AMERICAN UNIVERSITY OF BEIRUT

Resource Management for Integrated Cloud-Fog Network

by

Neam Mohammad Hassan Farroukh

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
July 2020

AMERICAN UNIVERSITY OF BEIRUT

Resource Management for Integrated Cloud-Fog Network by

Neam Mohammad Hassan Farroukh

Approved by:

Dr. Haidar Safa, Professor Computer Science	Advisor	Haidar Safa
Dr. Mohammad Nassar, Assistant Professor Computer Science	Member of Committee	[illegible]
Dr. Shady Elbassuoni, Assistant Professor Computer Science	Member of Committee	[illegible]

Date of thesis defense: June, 2020

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name:	Farroukh	Neam	Mohammad Hassan
	Last	First	Middle

[✓ Master's Thesis] Master's Project Doctoral Dissertation

[✓] I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☐ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One ___ year from the date of submission of my thesis, dissertation or project.**
Two ___ years from the date of submission of my thesis , dissertation or project.
Three ___ years from the date of submission of my thesis , dissertation or project.

N.F	7/2/2020
Signature	Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

First, I would like to thank my advisor Dr. Haidar Safa and the committee members, Dr. Shady Elbassuoni and Dr. Mohamad El Baker Nassar for their supervision throughout my thesis work to overcome the obstacles I have faced.

To my family and friends thank you for understanding and encouraging me, and most importantly absorbing my anger and breakdowns during my Masters' studies. You are my forever support system.

Moreover, I want to thank God for giving me the strength and keeping me sane despite what I went through to make this thesis perfect enough.

Looking back at what I have been through, it was not a smooth plot twist, but it was definitely worth the waiting. Thank you American University of Beirut for an unforgettable experience.

An Abstract of the Thesis of

Neam Mohammad Hassan Farroukh for Master of Science

Major: Computer Science

Title: Resource Management in Integrated Cloud-Fog Network

Fog computing extends the cloud services to the edge of the network by taking advantage of edge devices that have sufficient IoT resources (i.e, storage, compute, and bandwidth). "A cloud closer to the edge" has been proved as a promising solution for avoiding unbearable latency and network capacity saturation with the proliferation of IoT end-devices. Lately, researchers have noticed the impact of cloud-fog cooperation on the performance of the network in terms of latency, network capacity and security. While the cloud could handle heavy-weight delay-tolerant tasks, the fog becomes in charge of all light-weight delay-sensitive tasks.In such integrated networks, resource management becomes a key challenge that must be addressed effectively. Moreover, researchers have been preferring the clustered network topology for the fog layer over the flat one. In this thesis we aim to design and study two different resource management variations for fog network: flat and a clustered variations respectively. Both variations are formalized as an optimization problem in order to maximize the IoT tasks to fog assignments while satisfying not only the resources requirements of the issued tasks, but its QoS requirements as well. The comparison of both variations shows that the flat approach gives better results in terms of overall and fog delay when increasing the number of clusters in the topology, while the clustered approach results in lower number of tasks being rejected. Moreover, a baseline approach is presented as well to evaluate our proposed approaches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this Chapter we present a brief background of cloud computing and the reason behind deploying fog computing in real world's application. We describe the motivation and the problem that we aim to address. We then state our objectives and end with thesis plan.

1.1 Background

Computing in general is an on-demand model that allows users to benefit from services without worrying about how they will be delivered or where such services are hosted. In this context, cloud computing is an emerging technology defined as a tool that provides resources like CPU, I/O, and memory as a utility service based on the users' demands. Cloud computing (CC) has been widely deployed due to its benefits such as cost reduction due to its pay-as-you-go strategy, flexibility in capacity which allows scaling up and down the cloud capacity for end-users, and disaster management where backups are present to save the day in case of a crash or failure.

Lately, the number of smart end-devices, notable by the Internet of things (IoT), has been proliferating in a way where it is predicted to reach 50 billion devices by the year 2020. Consequently, such a number of devices is going to produce not less than 43 trillion gigabytes of data [2]. This huge amount of data, when sent to a far located cloud, causes network saturation and eventually degradation in users' experience. Thus, the bliss of cloud computing turns into an issue for latency-sensitive applications that require resources in the vicinity. To address this issue and to meet the delay and mobility requirements of various IoT applications, it was necessary to build a new control layer that resides between the end-devices and the far located cloud. This shifting from the core to the edge of the network is termed as edge computing. Based on this new research interest, Cisco proposed fog computing (FC) in 2012 [3]. According to Cisco, the Fog is a cloud closer to the IoT devices, where it extends the assets of the cloud as storage, computing and networking services to the edge of the network by taking advantage of devices, e.g access points, routers, rich in IoT resources located near end-devices, and thus resulting in lower latency and better users' experience.

1.2 Motivation

The network topology of the fog layer, as described in chapter 2, could be either flat where the fog layer consists of several fog nodes, or a clustered topology meaning that the fog layer consists of fogs (the clusters) each made up of several fog nodes. The main difference between both architectures is the number of fog controllers that are responsible for taking the task allocation decisions. The first type of topology has a single controller taking the decisions on behalf of every fog node present, while in the second type of topology, every fog has its own controller which is responsible for taking the decision for such a specific group of fog nodes belonging to a fog (cluster). Obviously, having several controllers would results in lower overall latency and network saturation and thus better performance measure. For that, recent research works in the fog field; such as in [1], [4],[5],[6],[7], have been interested in adopting the clustered architecture to apply their resource management approaches.

Resource management in fog computing is still considered as a key challenge due to the limited computational resources as edge devices have smaller processors and lower battery life than the cloud. Moreover, edge or fog devices are heterogeneous where each device has its processor and architecture. IoT applications are becoming more heterogenous and elastic as well, which means that each task issued has its QoS requirements, and its performance metrics differs based on the node (cloud or fog) it is assigned to.

1.3 Problem Definition

As the clustered architecture proposed, has its ability in optimizing the power of both the cloud and fogs in order to support the proliferation of IoT applications, it is essential to have a dynamic and effective resource management approach. Meaning that this challenge must be addressed in a way that optimizes the IoT resources while satisfying the QoS requirements of the issued IoT tasks.

Thus, the main problem that must be addressed is to find a secure resource management approach that has its ability in assigning IoT tasks, which require different resources such as CPU and storage and different QoS requirements such as allowed delay, to fog nodes which are capable of handling and satisfying such requirements.

1.4 Objective and Contribution Summary

In this thesis we propose a QoS and seucrity-aware resource management (QSRM) approach for a clustered integrated cloud-fog networks using two different variations. Both variations formalize the problem as a mixed-integer linear programming model (MILP). The first variations aims on examining all fog nodes that are reachable for the IoT device issuing the task in order to find the best node for task execution. While the second variation first finds the best reachable cluster and then works on selecting the best fog node among the nodes belonging to the selected cluster. The goal of both variations is to find the best fog node to execute any arriving IoT task based on the task's requirements and the node's capabilities in order to satisfy the QoS and security requirements while maximizing the limited resources utilization. Both variations will be able to evaluate the clustered architecture and the effect of scaling the topology up and down based on the average overall delay, average delay at the fog, the number of rejected tasks, and the number of tasks being offloaded to the cloud.
Briefly, our contribution can be summarized as follow:

- Present a clustered fog network topology.
- Formalize two different resource allocation variations for such a clustered topology, a flat and clustered based fog selection variations.
- Define the impact of adding the security and privacy metrics to our variations on the resource allocation efficiency
- Evaluate our variations by comparing them to a baseline approach based on the average overall delay, average delay at the fog layer, the number of rejected tasks, and the number of tasks being offloaded to the cloud.

1.5 Thesis Plan

In chapter 2, we describe cloud computing, fog computing, IoT definition and its applications, and clustered network topology. Moreover we will review work related to resource management in fog computing by stating their limitations and how their work differs from ours. In chapter 3, we describe the network architecture in addition to the profile vectors that define the tasks and the fog nodes. The possible scenarios, the proposed variations of our approach, and the methodology of calculating delay will be introduced as well in chapter 3. While in chapter 4 we define the simulator that has been used to implement our approach and present the obtained results for every variation. Both variations are compared against each other, and against a baseline approach. We conclude in chapter 5.

Chapter 2

Background and Related work

In this chapter we describe some basic concepts, such as introduction to cloud and fog computing, introduction to IoT, its applications and tasks' description, and clustered network topology. Moreover, we survey several related recent work found in the literature.

2.1 Introduction to Cloud Computing

Cloud computing is the replacement of local storage and servers. It enables customers and users to employ compute resources like applications, storage and virtual machines as a utility through the Internet without worrying about managing and maintaining them [8].
Cloud computing follows the Pay-as-You-Go usage model, which facilitates the scaling and customization of computing resources.
Clients, Data centers and Distributed Servers are the three main components that make up the cloud computing solution.
As virtualization is a process of manipulating the hardware, cloud computing refers to the service resulting from this manipulation. Virtualization is the foundation of cloud computing since it enforces scalability and flexibility.

The architecture of Cloud Computing is made up of four layers. The first layer is the hardware layer which deals with the physical assets of the cloud such as router, switches, servers cooling system and power. Second comes the the infrastructure layer, also known as the virtualization layer, which makes a pool of storage capacity and computing resources. The third layer will be the platform layer that comprises of operating systems and requisition structures. The last layer is the application layer which is made up of the actual cloud provisions.

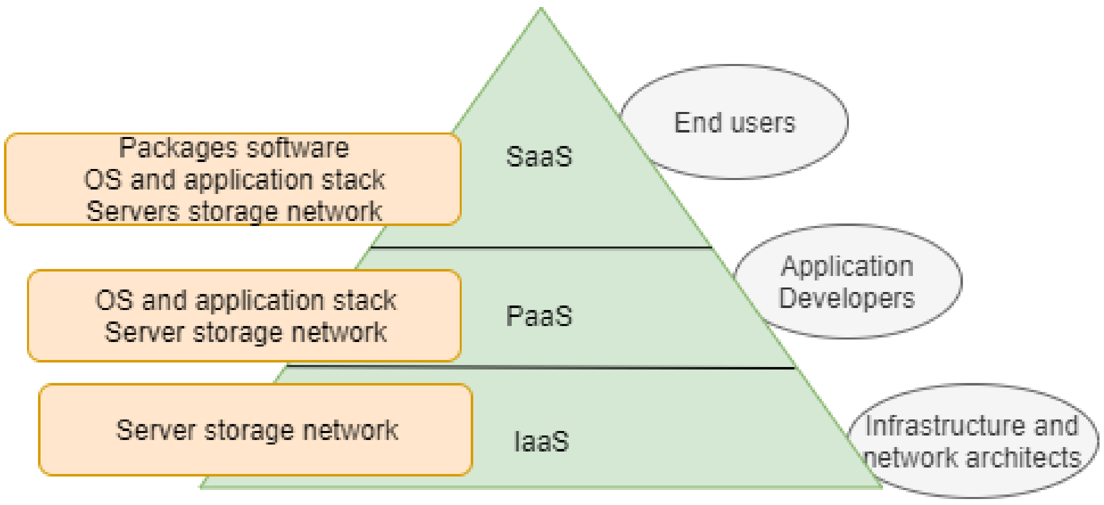


Figure 2.1: Cloud Computing Service Models

Cloud Computing has three different service models as shown in Figure 2.1 :

- Software as a Service (SaaS): Also known as on-demand software. SaaS replaces the traditional software installation, maintenance and management. It is a model where applications are hosted as a service to end-users that can access it through the internet.
- Platform as a Service (PaaS): It is a model that provides developers with all needed hardware and software tools to develop their applications directly on the cloud. This service model provides the cloud consumer with a development platform where they can focus on creating and running their appli-

cations instead of constructing and maintaining the infrastructure. Using this model the development team can collaborate and work together despite of their physical locations.

- Infrastructure as a Service (IaaS): Instead of offering applications as the previous two models, this model provides the consumer with the hardware that the organization can use to store whatever they want in it. Rather than having the organizations purchasing servers and having to pay for the datacenter space for them they can just rent these resources on the cloud. The model is scalable and supports multi-tenancy.

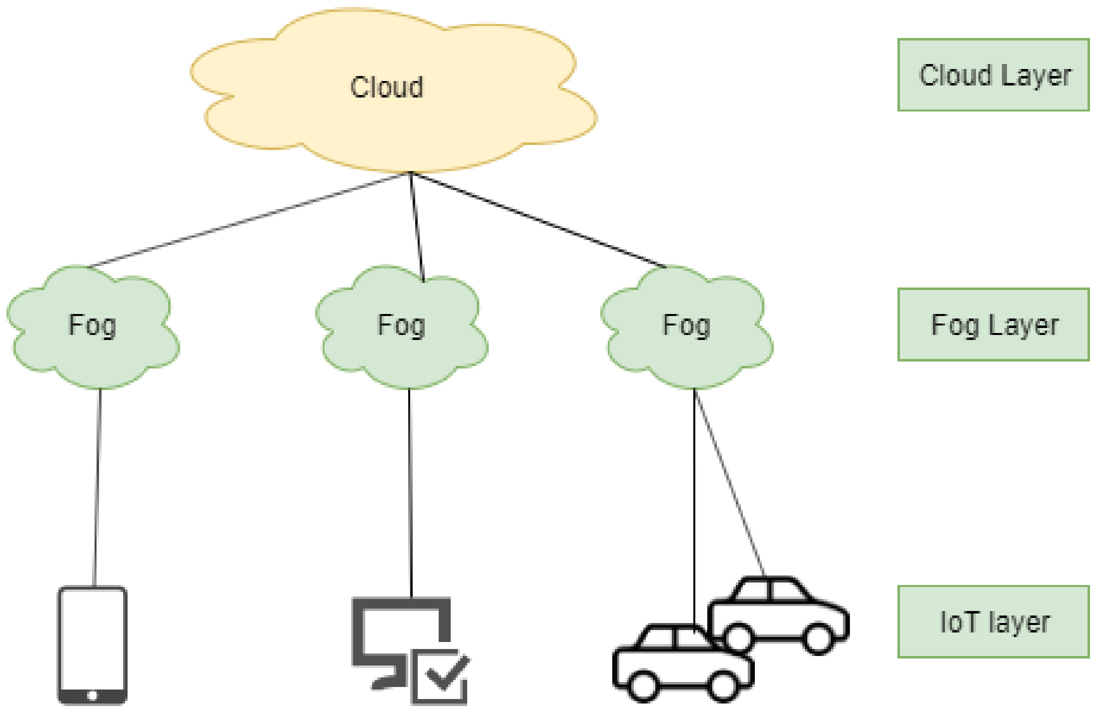


Figure 2.2: Fog Computing General Architecture

2.2 Introduction to Fog Computing

2.2.1 Fog Computing Definition

Fog computing is allowing the computing to take place at edge of network on behalf of the cloud and IoT services. The edge is any computing and network resource that resides between the end-users and the cloud data centers as shown in Figure 2.2. The main purpose behind fog computing is to perform all the computing at the proximity of the data sources. Besides the computing performance, fog computing performs data storage, caching and processing. Moreover, it can distribute requests and deliver services from the cloud to users.

2.2.2 Aim behind Fog Computing

As the amount of data being generated at the edge of the network is growing fast due to the proliferation of IoT devices, response time will massively increase if data processing takes place at the cloud due to the current bandwidth capabilities and network saturation [9]. The aim behind fog computing, is to perform all the processing and computing at the proximity of data resources and thus minimizing the latency. Fog nodes i.e routers, switches, and access points are also able to support mobility due to their dense geographical distribution. As the fog extends the cloud to the edge of the network, it depends on the existence of the cloud and can not operate in a standalone manner. Starting from this, the idea of integrated cloud-fog networks emerged which allows applications to span over the cloud and the edge.

2.3 IoT Definition and Applications

2.3.1 IoT Definition

Any device that is able to transmit and receive data, or has a sensor attached to it is known as an IoT device. IoT devices include computer devices and wireless sensors. Inter connectivity, heterogeneity, dynamic changes, enormous scale, sensing, and intelligence are the most important characteristics of IoT devices. These devices have specific requirements such as performance, availability, scalability, and serviceability.

2.3.2 IoT Applications and Tasks Examples

IoT devices are becoming part of every aspect of our lives since they give more control on routine life work and personal tasks. For that, IoT applications have been deployed in various areas such as smart homes, smart cities, transportation, and healthcare . To start with smart homes, IoT applications are used to have more control over simple tasks in our homes. Examples of such applications are door locks, smart heating, smart gardening and personal assistants. As for the healthcare area, IoT devices can be used to save lives. For example IoT devices can be used to continuously monitor glucose and insulin pens for diabetic patients. Moreover, new devices are introduced for coagulation testing. These devices help patients to detect how quickly their blood clots in order to avoid strokes or bleeding. For the smart cities area, IoT applications can be used to provide an innovative solution for traffic congestion. For example, smart traffic signals can adjust their timing in peak hours to keep cars moving, or city officials can collect data from traffic cameras and road sensors to monitor traffic incidents in real-time like alerting drivers in a way to direct them to less congested routes.

To go deep into describing IoT tasks in a more technical way, in smart traffic signals, the sensors on the road (acting as the IoT devices) collect data such as the number of vehicles present and send it to the fog nodes or the cloud. The latter will analyze this information and return its response back to the traffic signals which have to adjust their timing in a way to keep the cars moving. Another example will be the asthma monitoring device that detects several symptoms as heart beats, cough rate, respiration pattern and send them to the fog node for analysis to be able to notify the patient in case of deviation from individual norm. Thus, any IoT task will be represented by data collected from IoT devices having specific resources and QoS requirements for execution. It is then sent to the fog node; which must be selected by applying a task allocation algorithm, for execution and analysis to return the desired output back to the device issuing the task.

2.4 Clustered Fog Network Topology

The aim behind a clustered fog network topology is to decrease the network saturation and overall latency. Many research work proposed such an architecture in order to tackle the resource management challenge. For example in [4], the authors mentioned that CC and FC are not mutually exclusive, rather they are interdependent. Meaning that the cloud can manage the fog and handle the heavy-weight delay-tolerant IoT tasks, while the fog can be responsible for handling the light-weight delay-sensitive IoT tasks. They proposed the iCloudFog, an integrated Cloud-Fog architecture which is meant to provision resources of the cloud or the fog to the IoT requests based on their availability and the requests' requirements. This proposed architecture is a clustered one, where the fog layer consists of fogs each made up of a fog controller and fog nodes. The authors pointed out the challenges that must be addressed while constructing such an integrated architecture which are network dimensioning which decides the size of each fog, Qos security-aware resource management, and localization.

Similar to [4],in [1, 6, 7] the authors adopted the same clustered fog layer for applying their proposed resource management approach. The fog layer in the all mentioned papers, consist of fogs. Each fog contains micro data centers, base stations and/or routers acting as fog nodes to execute tasks issued by IoT devices.

2.5 Related Work

In this section we review the contributions of several authors in addressing the resource management challenge in a fog computing environment.

To improve the performance of fog networks in terms of throughput and cost, an offline priority-based task-scheduling algorithm was proposed in [10]. In this algorithm the resource allocation for task execution is a two steps process. In the first step, requests will be marked as accepted or rejected by the nearest fog node's manager. When a request is accepted, meaning that the time it will spend in the fog layer is less than its maximum allowed delay, it has to be prioritized first before assigning it to any fog node. The request will be added to one of the three priority queues, high, medium, or low based on three factors, its maximum allowed delay, estimated service time and original labeled priority queue. As the prioritizing step is over, the second step which is the resource allocation process starts. The requests will be assigned to fog nodes and processed based on the priority queues they belong to. The proposed network allows a fog node to either execute the arriving tasks locally if its resources are sufficient, reallocate the task to another fog node in case there are sufficient resources in the fog layer following step one again, or send the task to the cloud for execution when all the resources in fog layer are saturated. Although the authors took into consideration the delay constraints and resources availability while assigning requests to fog nodes, this offline approach cannot cope with the elasticity of requests where each task to node assignment would result in a different performance and thus it is becoming infeasible to have a prior knowledge of the applications.

A service oriented resource management approach, which depends on service cost optimization, was proposed in [11]. In this architecture, the fog node, which is responsible for providing on-demand services to end-users, is also capable of predicting their consumption of resources. To allocate and consume resources in a fair and effective way, the proposed predictive model is set based on the users' behavior and their probability of relinquishing or giving up on resources they just requested. The fog layer keeps track of historical records for all existing customers which includes their service relinquish probability for every service they have requested and the average overall relinquish probability that describes the overall users' behavior and loyalty on all the requested services. The resource allocation process is for the three types of customers: a new customer, an existing customer, and an existing customer requesting a service for the first time. This resource allocation scheme assigns a virtual resource value to the customer based on its type, and/or behavior, and the service cost, then this value in return is mapped to the actual resource pool based on the type of service. This approach, which treats end-users differently to optimize the resource utilization and service cost, is able to resolve the task offloading challenge without taking into consideration the complexity of the network, load balancing the latency constraint which will result in degradation in users' experience and QoS satisfaction.

Distributed earliest deadline first for fog (DEF) algorithm was proposed in [12] to allocate resources in a cellular fog computing architecture. In this architecture, each cell consists of a single fog service provider called an aggregator and a set of edge devices all connected to the aggregator. The aggregator acts as the controller that keeps track of all the service demands and resource capacities of the devices and runs the real-time scheduler. The edge devices can act as a service consumers or service providers. Due to its mobility and upon the arrival to the system, the service consumer, a client, should provide its arrival time, task deadline, and its length as parameters to the aggregator while the service providers, hosts, advertise their arrival time, departure time, and computation power in million instructions per second to the aggregator. The aim of DEF algorithm is to maximize the utilization of residual computation capacities of end-devices. As clients arrive with tasks, the aggregator will prioritize the tasks based on their deadline, where the highest priority is given to task with earliest deadline. The tasks are mapped to computing devices under the same aggregator based on best fit method. A device is selected as a best fit for a task if it has the least remaining computation capacity after executing the task within its deadline. If the algorithm can't find a best fit then the task is rejected. Thus to find the best fit device, the computation capacity required by the task should be less than or equal to the computation capacity available at the device within the tasks deadline. This algorithm is extended in order to guarantee minimum amount of profit to aggregator. The aim is to meet the constraints on task budget and computation rate, which is a price charged by hosts for every task scheduled on it. Here, a best fit device is selected if the computation capacity required by the task is less than or equal to the computation capacity available at the device within the tasks deadline and the task budget is greater than or equal to the sum of the aggregator's minimum profit and amount paid by task to device for execution. Although this approach deals with delay constraint, the possibility for a task to be rejected is high as the load balancing between aggregators, and the offloading to the cloud are not taken into consideration.

The fog resource selection algorithm (FResS) was proposed in [5]. The algorithm consists of three main modules which are: task scheduler, history analyzer, and resource selector. The IoT requests are submitted to the task scheduler, which acts as an interface for IoT devices, fog, and cloud layers. The task will be sent to the history analyzer, which consists of artificial neural networks, to predict the resources that would be capable of handling the task and the possible execution time. The predictions are then sent to the resource selector to find the suitable fog node to execute the task. The resource selector discovers and tests fog resources against the minimum criteria such as their availability, operating system, cost and delay constraints. The task scheduler then submits the final decision of the resources needed with the task to the fog layer for execution. The predictions made are stored in a separate database to be used as training vectors for future predictions. These three modules help improving the performance of IoT devices that communicate directly with the fog layer and causing high delay and network saturation. This approach suffers from a main limitation which is the availability of previous observations in order to predict the resources needed for future IoT requests.

Machine learning was proposed for resource allocation at the edge of network in [13] as well. This solution assumed the tasks are elastic, meaning there are no specific resource requirements, and that each task to node assignment would result in a different performance measure that should be learned from previous observations of the task to node assignments. Hence assigning a task to a node is simply mapping a pair of task and node's profile to a performance metric. The assignment problem is viewed as a recommendation system. Multiple nodes can be recommended for a task based on the performance of previous similar observations of assignments. The predictions are done using a collaborative filtering (CF) network which takes as input the vectors describing the tasks requirements and edge nodes capabilities and output the predictive performance vector which could be mapped to a single value by a utility layer. This task to node assignment architecture proposed consists of three components. The first component is the resource and performance monitor (RPM) which measures the performance profile of a task as it is assigned to a node. In addition to that, it monitors the currently available resource profile of each node. The measured performance with the updated information about the node's status is stored in a database and then fed to the other two components. The second component is the predictive performance modeler (PPM) which is responsible for building and re-building the CF-based performance prediction model using the data provided by the RPM. The model is updated periodically within fixed period of time or on the arrival of n-th new vectors. The PPM then provides a recommended set of nodes for a given task to the Task-Resource Scheduler which is the last component that is responsible for assigning a task to one of the nodes in the recommended set while taking into consideration the current resource profile of the nodes.

The sparsity of available observations of IoT tasks being executed on fog nodes can be seen as a challenge that [5] and [13] suffer from. As these approaches need the observation of nodes' performance in order to predict the needed resources for future tasks.

The load balancing challenge in software defined networking (SDN) based fog networks was addressed in [14] using Q-learning from reinforcement learning to achieve the minimum communication and task processing delays. The architecture consists of an SDN-fog controller and serving SDN-fog nodes. The SDN-fog controller is responsible for observing the environment by collecting information, and taking the decision on behalf of every fog node. Whereas the fog nodes serve the end-users and deliver the collected information to the controller. As any reinforcement learning algorithm, the problem is formulated as a Markov decision process. The node having requests to be allocated, the number of tasks need to be allocated, and the number of tasks currently in the queue, represent the state vector. While the action vector is represented by the node the tasks will be offloaded to and the number of tasks to be offloaded. The main goal is to distribute the incoming workload across a group of serving fog nodes to maximize the utility while minimizing the processing time and the overall overloading probability.

For that, they designed their reward function in terms of the utility, immediate delay, and overloading probability functions. This approach may cause network saturation as the decision should be taken by only one controller.

Several reinforcement learning algorithms such as Q-learning, SARSA and E-SARSA were used in [15] to address the resource allocation challenge in fog radio access networks, in order to meet the QoS requirements as latency and throughput of IoT tasks . The network architecture consists of three layers. The core layer includes the cloud. The fog layer consists of fog nodes serving the IoT devices in the IoT layer. Each user request has a utility value measured using a function of the latency requirements, throughput requirement, and channel capacity. The authors defined the state of a fog node at any given time as a function in terms of the utility value of the request and the number of available resource blocks. The fog node can choose between two actions, either accept which means execute the task locally, or reject meaning to offload the task to the cloud. Four values represent the reward for every possible action. A threshold value for the utility value is introduced to facilitate learning the expected future reward. This approach does not take into consideration the latency of the task when offloaded to the cloud which can be addressed through load balancing.

A conceptual framework for fog resource provisioning which aims at minimizing the delay and maximizing the resource utilization was introduces in [1] .The architecture consists of three layers IoT, fog, and the cloud shown in Figure 2.3.

To allow the orchestration and resource provisioning in both cloud and fog the framework introduced the cloud-fog control middleware that resides between the fog and cloud layer. This control level is the central unit that manages the execution of tasks in the cloud while supporting the underlying layers. Moreover, as the fog applications can run without the cloud involvement, the framework introduced another control level for the fog layer, which are fog orchestration

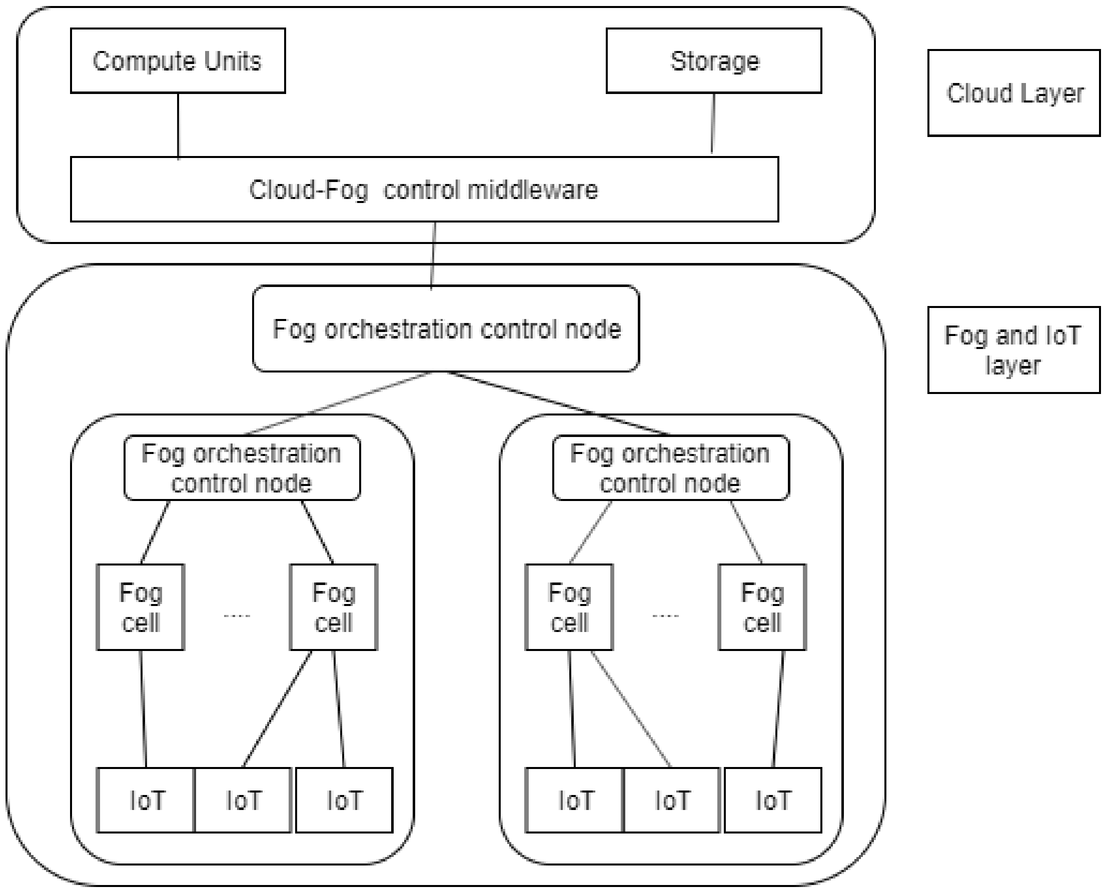

Compute Units
Storage
Cloud Layer
Cloud-Fog control middleware
Fog orchestration control node
Fog and IoT layer
Fog orchestration control node
Fog orchestration control node
Fog cell
....
Fog cell
Fog cell
....
Fog cell
IoT
IoT
IoT
IoT
IoT
IoT

Figure 2.3: Network topology in [1]

control nodes. These nodes are responsible for managing the fog colonies connected to it. Each fog colony is made up of a single fog orchestration control node and a number of fog cells. Each fog orchestration control node is responsible for performing changes in a fog colony, creating resource provisioning plan for task requests, and monitoring fog cells and IoT devices. The resource provisioning plan generated by the orchestration node is formalized as an optimization problem. As mentioned previously, each fog colony consists of a control node and a set of fog cells, each node and cell is represented by the CPU and RAM parameters. Moreover, every control node and cell has a type which indicates the set of services that can run on this particular cell. The links between a control node and a cell is characterized by the bandwidth and delay. The tasks arriving to a fog colony are represented by the CPU and RAM demands and its type. If a particular fog colony can't execute the arriving task due to lack of resources, this task is then propagated to higher levels where it can be allocated to other fog colonies or propagated to the cloud. In order to maximize resource utilization as bandwidth and to minimize the delay caused by propagating tasks to cloud, the aim of the proposed optimization problem is to maximize the number of task to cells assignment while decreasing the propagation of tasks to higher level which is formulated as follow:

$$\text{maximize} \sum_{i=1}^{M} (\sum_{j=1}^{N} (x_i^j) + x_i^F)$$

where N is the number of requests M is the number of fog nodes. The decision variable x_i^j is set to 1 if $request_i$ is assigned to fog $node_j$ and 0 otherwise.
The first constraint for this objective function is that the type of the request should conform to the type of the fog cell. Moreover, as a second constraint, each request can be assigned exactly to one fog cell. To minimize the delay, fog cells are sorted according to the value of delays, which means prioritizing the fog cells with lower delay and then selecting a fog node with minimum estimated delay. Two more constraints were added to insure that the tasks do not exceed the resources of fog cells. This approach is similar in concept to what we are aiming to achieve yet the hierarchy and network topology differs. Moreover, this approach is not a security aware one.

2.6 Discussion

In our work we aim on adopting the clustered fog network topology described in section 2.4. Table 2.1 summarizes some of the surveyed related work. it shows features such as privacy and security, latency and load balancing. It also shows the features we aim to address in our proposed work. As we can see, none of the related work tackles the security problem, which means selecting a fog node that satisfies the security and privacy required by the issued task. In our proposed work we aim on providing a run time resource allocation approach for such a complex and heterogeneous integrated network with sparsity of available observations as in [1], [14], and [15]. As mixed integer linear programming is an applicable solution for real world applications in allocation and planning problems, we aim at applying it to formalize the QoS and security-aware resource management approach for integrated cloud-fog networks. We will propose two different variations of the same approach as follow:

- The first variation: The controller of the fog (cluster), have full knowledge about the network and the fog nodes reachable for the devices connected to it therefore the controller first will receive the task issued from the IoT device and then will examine all the reachable fog nodes using the formalized optimization problem to select the best node to execute the task.
- The second variation: The controller of the fog (cluster) also having a full knowledge about the network, after receiving the task, will first find the best cluster for execution and then examines the nodes of that cluster to select the best node for task execution

In order to evaluate the clustered topology, both approaches will be implemented and then evaluated using different scales of; i.e, we will scale the topology down by adding few clusters and up by increasing the number of clusters. We will measure the average overall delay, average delay at the fog, the average number of rejected tasks and the number of tasks being offloaded to the cloud, the obtained results will allow us to determine the best topology scale.

For that, the formalized optimization problems will be able to address the three issues delay, security, and load balancing, to point out which approach and topology scale yields better results in terms of throughput, latency and security.

Table 2.1: Comparing several authors contributions and our proposed work

	Addresses		
	Delay	Load Balancing	Security
[5]	✓	✓	
[1]	✓	✓	
[10]	✓	✓	
[11]			
[12]	✓		
[13]	✓	✓	
[14]	✓	✓	
[15]	✓		
QSRM	✓	✓	✓

Chapter 3

System Model

In this chapter we present our proposed QoS and security aware resource management approach for an integrated cloud fog network and describe its network architecture and possible deployment scenarios. We also formalize it as an optimization problem.

3.1 Network Architecture

In this thesis, we aim on evaluating the clustered fog network architecture by taking into consideration two different task allocation variations for it. For that, we adopt a network architecture similar to the one proposed in [4]. The architecture consists of three layers: the cloud, the fog, and the IoT. The cloud layer consists of the cloud data centers. The fog layer consists of fogs representing the clusters, each made up of multiple fog nodes. The IoT layer is made up of all the IoT devices requesting services from the upper layers. Four types of communications could take place: cloud to fog (C2F), fog to fog (F2F), fog to thing (F2T), or thing to cloud (T2C). It is essential to have small scale fogs in order to deal with data locality while supporting mobility. The fog layer is heterogeneous in nature as the fog nodes could be either wireless, wired or hybrid. In this integrated

Table 3.1: Table of Notations

Notation	Definition
f_j	fog node j
t_i	IoT task i
CPU_j,Memory_j	CPU and Memory capability of fog node respectively j
IPS	Processor's speed of of fog node j (instruction per second)
PM_j	Privacy Measure of fog node j
CPU_i,Memory_i	CPU and Memory required by IoT task respectively i
SL_i	Security level required by IoT task i
Bytes	Size of IoT task i in Bytes
MAD	Maximum allowed delay for IoT task i
SC_i	Scheduling class of IoT task i (high or low)
$\mathrm{BW}_{s,d}$	Bandwidth between source s and destination d
$\mathrm{PD}_{s,d}$	Propagation delay between source s and destination d
X_i^j,Y_i^k	Decision variables for optimization model
$\mathrm{TotalDelay}_i^j$,	Total delay of executing task i on node j
$\mathrm{SpareTime}_i^j$	Difference between maximum allowed delay by task i and $\mathrm{TotalDelay}_i^j$
$\mathrm{Privacy}_i^j$	Difference between security level required by task i and privacy measure of node j
$\mathrm{TotalDelay}_i^k$,	Total delay of executing task i in cluster k
$\mathrm{SpareTime}_i^k$	Difference between maximum allowed delay by task i and $\mathrm{TotalDelay}_i^k$
$\mathrm{Privacy}_i^j$	Difference between security level required by task i and average privacy measure of cluster k
AvgCPUk,AvgMemory$k$	Average CPU and Memory available at cluster k respectively
RD_{atC_j}	Request delay at controller of fog node j
Delay_{atf_j}	Delay of executing task on fog node j
α	Hyper-parameter

network, the fog nodes can be shared with the cloud to allow distributed storage and CPU. Moreover, IoT devices can act as fog nodes as long as their resources are sufficient to handle light-weight delay-sensitive tasks. A heavy-weight delay-tolerant task will be offloaded to cloud for processing. Each fog node is associated with a controller that keeps track of the fog nodes' resources and handles the F2F communications. The T2C communication could take place through the fog layer where the fog nodes will act as routers.

In our proposed approach we assume that the network architecture consists of three layers as shown in Figure 3.1: the the cloud layer holding the cloud servers, IoT layer containing all the IoT devices issuing tasks, and the fog layer consisting of a number of small scale fogs (clusters).

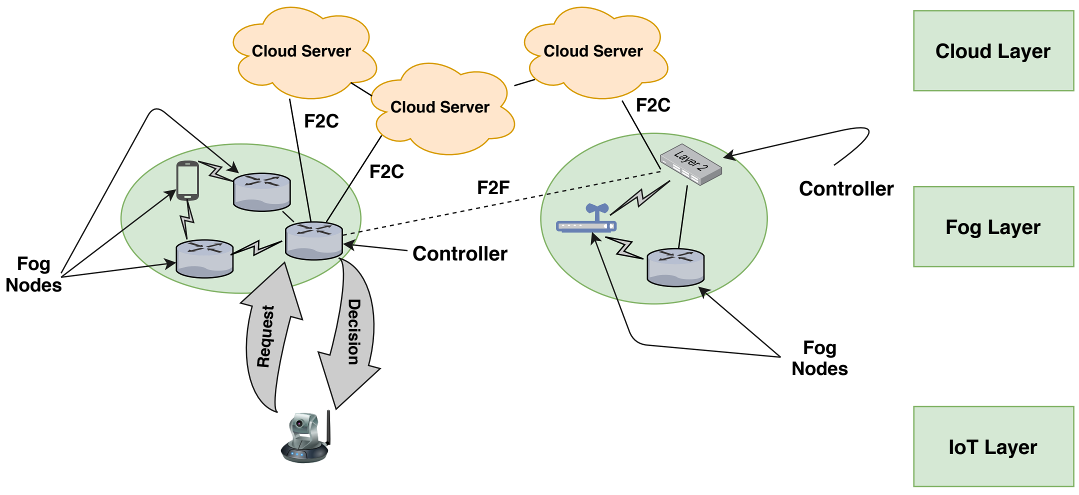


Figure 3.1: Network Architecture

The number of fog clusters might vary. Each fog cluster contains a number of fog nodes with a single fog controller to take the task allocation decisions. Each fog cluster will have a number of IoT devices from the IoT layer directly connected to it, meaning that the tasks issued from any IoT device will be directly transmitted to the controller of the cluster it is connected to. Therefore, an IoT device could reach one or more fog clusters, but it will be directly connected to a single fog cluster. As mentioned in [4], fog controllers could communicate through F2F communication in order to allow load distribution and the ability to execute tasks not only in the fog clusters the device connected to, but in the reachable fog clusters as well. Moreover, the fog nodes could act as routers to propagate tasks to the cloud when having heavy weight delay-tolerant tasks.

3.2 Profile Vectors of Fog Nodes and Tasks

As mentioned previously, the aim of this thesis is to propose a QoS and security aware resource management approach for an integrated cloud-fog network.Table 3.1 srates the definition for all notations used in this chapter. Each fog node f_j will be featured by a vector { CPU_j, $Memory_j$, IPS, PM_j } represented by its available CPU, available storage, instructions per second, and its privacy measure which is a trust value representing the security strength of fog nodes [16], respectively. Each task t_i arriving at the fog layer is featured by a profile vector { CPU_i, $Memory_i$, SL_i, IC, $Bytes$, MAD, SC_i } as well. This vector consists of the required CPU, memory, security level, instructions count, its size in bytes, its maximum allowed delay , and the scheduling class respectively. The scheduling class for the task t_i given in equation 3.1 is used to tell whether or not it is delay sensitive,

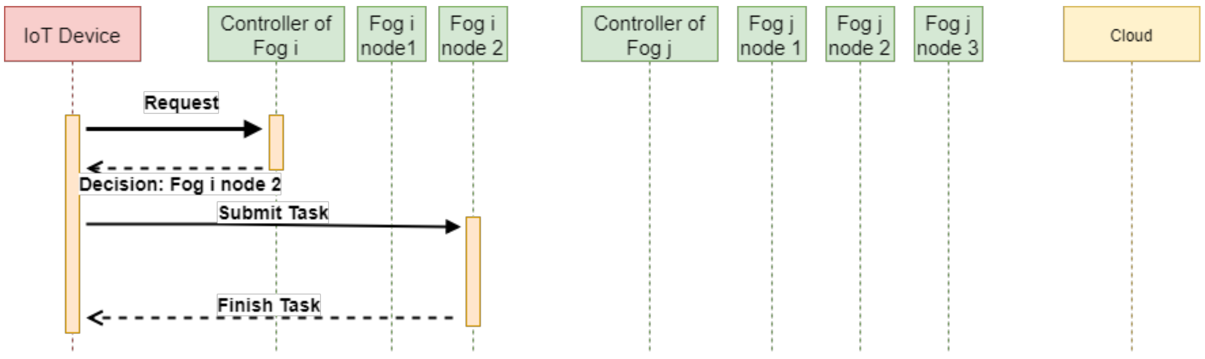
$$SC_i = \begin{cases} 1, & \text{if } Task_i \text{ is delay tolerant} \\ 0, & \text{Otherwise} \end{cases} \tag{3.1}$$

The security level attribute will help the controller to decide if a task can tolerate the delay in case it is offloaded to the cloud.

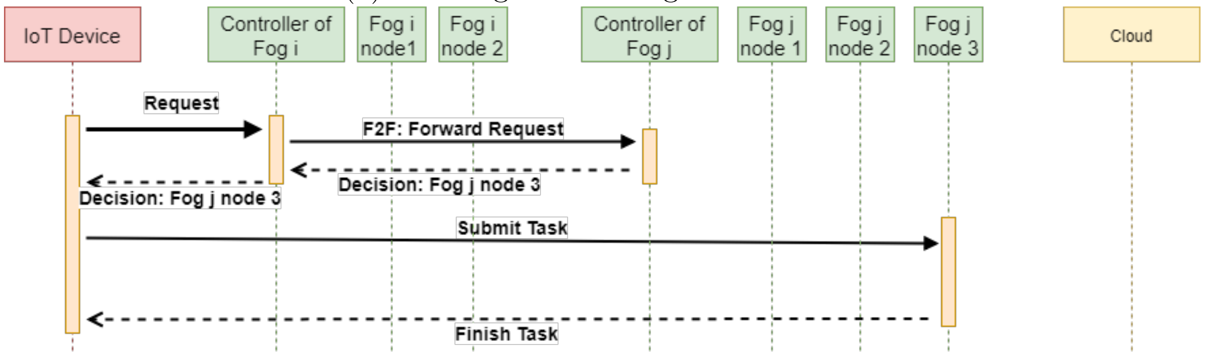
The privacy measure of the fog node is a trust value. Each task will have a security level class which is mapped to a range of privacy measures. The fog node must be able to satisfy the minimum privacy measure required by the task. For example, assume we have a fog node f$_j$ having a privacy measure 0.8 and a task i with security level class 'A' which is mapped to a range between 0.6 and 0.9, then we can say that f_j can satisfy the minimum security level required by i which is 0.6 and thus it can be considered as an option for its execution from the privacy perspective.
As for the memory and CPU, a fog node is considered as an option for a task execution if the CPU and memory required by the task do not exceed the available CPU and memory at the fog node.
The maximum allowed delay attribute is used to satisfy the latency requirement of the task. Thus a node is applicable for execution if the total time it takes to execute the task along with the total transmission and propagation delay do not exceed the maximum allowed delay attribute of the task.
The link between any two network devices source s and destination d (as IoT devices, fog nodes and cloud) is represented by the vector $\{BW_{s,d},\ PD_{s,d}\}$ stating the bandwidth and propagation delay respectively. This vector represents the communication and the network topology and states which devices are connected to each others. Moreover, the vectors attributes of every link will be used to calculate the propagation and transmission delay when taking the task allocation decisions.

3.3 Possible Scenarios

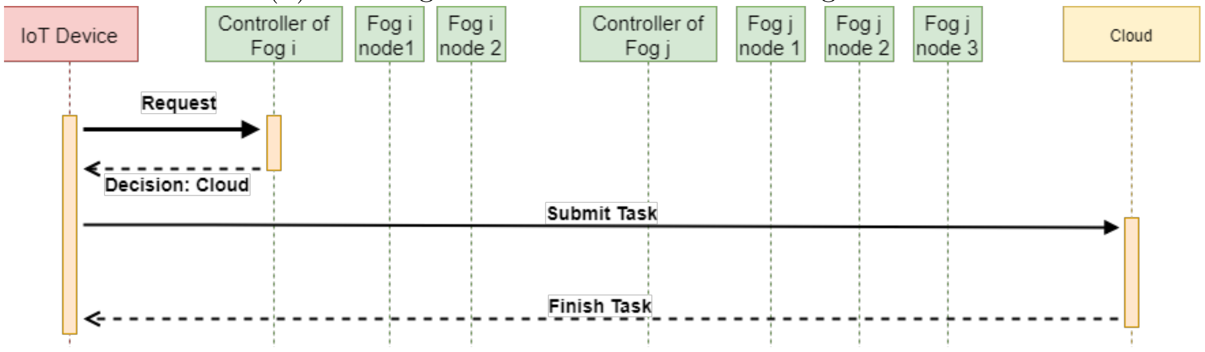
The requests issued from the IoT devices will be submitted to the fog controller of the nearest fog. The controller, who has the full knowledge about the profile vectors of the devices connected to it, its fog nodes, and the reachable neighbor fog nodes, will be able to decide which fog node is suitable to execute the sub-



(a)Choosing a Local Fog Node



(b)Choosing a Local Node in another Fog



(c)Offloading Task to Cloud

Figure 3.2: Three Possible Scenarios

mitted task.

The controller receives a request from an IoT device directly connected to it, the controller's job is to find the node which is able to handle the task represented by the request. The decision taken will be sent back to the IoT device issuing the request in order to redirect it to the selected node to submit the task directly. Thus, the controller has the ability to either select a local fog node as in Figure 3.2 (a), a node in other reachable fogs as in Figure 3.2 (b), or offload the task to the cloud as in Figure 3.2 (c). The controller can always reject the task if none of the available resources can meet the required requirements.

Figure 3.2 shows that any IoT application in our proposed approaches is composed of four messages. The first message represents the task request issued from the IoT device to the controller of the fog. The second message holds the controller's decision regarding the message request, meaning that the controller can either reply with the selected fog node which the IoT device has to submit the task to, inform the IoT device that it has to offload its task to the cloud, or inform it that its task was rejected. The third and fourth messages will be issued in case a fog node was selected or the task is going to be offloaded to the cloud. Thus, the third message will be the message used to submit the task from the IoT device to the selected fog node or the cloud, while the fourth message will be the reply from the node which executed the task back to the IoT device.

3.4 Two Variations of the Proposed Approach

In this section we propose two variations of our approach that the controller can follow to be able to take a decision and select the best node; if applicable; to execute an IoT task. In the first variation, the controller can select the best fog while in the second variation the controller can select the best fog cluster.

3.4.1 First Variation: Flat Based Fog Selection

In this variation the controller has the full knowledge about its fog nodes, the devices connected to it, and the fog nodes these devices can reach. It has to examine all the fog nodes; which are considered as an option for selection, based on an optimization formulation to come up with a final decision. The optimization model is a mathematical model that aims on finding values for the decision variables that maximize or minimize an objective function without violating some constraints.

Thus, an IoT device first issues a request of Task T_i which holds the resources and QoS requirements of the task to the controller of the fog cluster it is directly connected to. The controller has to either select the best node if found, choose the cloud as an option if the task can tolerate a long delay based on its scheduling class described in section 3.2, or reject the task if none of the previous two options is applicable. The controller then has to notify the IoT device with its decision.

The Decision Variables

We define X_i^j as the decision variable for our optimization problem. If X_i^j is 1, this means that Task T_i is assigned to fog node f_j and 0 otherwise as given in equation 3.2.

$$X_i^j = \begin{cases} 1, & \text{if Task } T_i \text{ is assigned to } f_j \\ 0, & \text{Otherwise} \end{cases} \tag{3.2}$$

When all the decision variables are zeros, it means that none of the reachable fog nodes is suitable for executing the task. In such a case, based on the scheduling class of the task, if it is set to 1, meaning that it is delay tolerant the decision will be to offload the task to the cloud, otherwise the task will be rejected.
Thus, our optimization model has to find the described decision variable which

indicates that our problem is a mixed-integer one, as the values that we aim to find are binary ones.

Add to that, the controller has to take a decision that does not violate the delay and privacy constraints of the task. As the controller knows the IoT device requesting a task execution, it can know which fog nodes can be checked if are suitable for the execution.

Delay Constraint

The whole task execution process starts first by the IoT device issuing the task request to the controller, and then the controller replying with its decision. The controller has to alert the IoT device about whether its task has been rejected due to unavailable resources, or accepted and thus notifying it about the node (cloud or fog) that has been selected for execution. As the IoT device is notified, if its task request is accepted, it will submit the task to the selected node (cloud or fog) and wait for the response and the result of the execution. As we mentioned in section 3.2, each task has the maximum allowed delay attribute which describes the maximum delay it can tolerate. Thus when a controller chooses a fog node as the selected node for execution it has to make sure that the total end-to-end delay between issuing the request of task T_i and performing the task on the selected node f_j ($TotalDelay_i^j$ in equation 3.3) should not exceed the maximum allowed delay of the task. Meaning that the difference between both values in equation 3.3 should be greater than zero as in equation 3.4.

$$SpareTime_i^j = MaximumAllowedDelay_i - TotalDelay_i^j \tag{3.3}$$

$$SpareTime_i^j \geq \mathbf{0} \tag{3.4}$$

In section 3.5 we will be describing how to calculate the $TotalDelay_i^j$ variable mentioned in equation 3.3.

In order to minimize the end-to-end delay we need to maximize the difference between the allowed delay and the total delay described in equation 3.3.

Privacy Constraint

The controller must select a fog node that guarantees the minimum security level required by the task and thus not violating the privacy constraint of the issued task request. This means that the difference between the minimum required security level of the task T_i and the privacy measure of the fog node f_j as in equation 3.5 should be greater than zero as in equation 3.6.

$$Privacy_i^j = SL_i - PM_j \tag{3.5}$$

$$Privacy_i^j \geq \mathbf{0} \tag{3.6}$$

Similar to what have been discussed regarding the delay constraint, we need to maximize the privacy difference as well to guarantee that the selected fog node satisfies the security level required by the task

Objective Function

Putting all of this together, our aim is to either assign a task T_i to a fog node f_j if is suitable, offload it to the cloud if it is delay tolerant, or reject the task if none of the previous options is applicable. For that, our objective function is to check if there exit a node that maximizes the sum of the factors computed using equations 3.3 and 3.5.

Our objective function is formulated as follow:

$$\text{maximize} \sum_{j=1}^{M} (\alpha SpareTime_i^j + (1-\alpha) Privacy_i^j) * X_i^j \quad \forall\, i \in \{1, \ldots, N\} \tag{3.7}$$

The α in equation 3.7 is a hyper-parameter used to give a weight for one factor over the other. This value will be set during simulation.

This objective function is subjected to several constraints listed below. As a first constraint, each task can be assigned to exactly one fog node. This constraint is formalized in equation 3.8:

$$\sum_{j=1}^{M} X_i^j \leq 1, \quad \forall\, i \in \{1, \ldots, N\} \tag{3.8}$$

Next, the selection should guarantee equations 3.4 and 3.6 as well. For that we added the two constraints presented in equations 3.9 and 3.10 respectively:

$$\sum_{j=1}^{M} (SpareTime_i^j * X_i^j) \geq 0 \;, \quad \forall\, i \in \{1, \ldots, N\} \tag{3.9}$$

$$\sum_{j=1}^{M} (Privacy_i^j * X_i^j) \geq 0 \;, \quad \forall\, i \in \{1, \ldots, N\} \tag{3.10}$$

Resources Constraints

Although our aim is to satisfy the delay and privacy constraints of the IoT tasks, yet our variation needs also to meet the resources requirements as in CPU and memory of the tasks. This means that when the controller selects a fog node to perform a specific task, it has to make sure that the resources required by the task do not exceed the resources available at the selected fog node.
For this purpose, equations 3.11 and 3.12 were added as constraints to assure that

the resources requirements; which are the CPU and the memory, were satisfied without exceeding the fog resources.

$$\sum_{j=1}^{M}(CPUj - CPUR_i) * X_i^j \ \geq 0\ ,\ \ \forall\, i \in \{1,\ldots,N\} \tag{3.11}$$

$$\sum_{j=1}^{M}(Memoryj - MemoryR_i) * X_i^j \ \geq 0\ ,\ \ \forall\, i \in \{1,\ldots,N\} \tag{3.12}$$

Where CPU_j and $Memory_j$ represent the CPU and memory available at fog node f_j, while $CPUR_i$ and $MemoryR_i$ represent the CPU and memory required by the task T_i

3.4.2 Second Variation: Clustered Based Fog Selection

In the first variation we assumed that the controller having the full knowledge about the fog nodes each of its IoT devices connected to, has to check if any of those fog nodes is suitable to execute the task requested by the IoT device by applying the optimization model described in the previous section. In this section, as a second variation, in order to take advantage of the clustered topology, we will allow the controller to first select the best cluster instead of directly searching for the best node.

For this purpose, we are going to add a new feature vector for each fog cluster k $\{$ $AvgCPU_k$, $Avg\ Memory_k$, $AvgIPS$, $AvgPM_k$ $\}$. This vector specifies the cluster's average available CPU, average available memory, average instructions per second, and average privacy measure respectively. These values are computed based on the profile vectors of the fog nodes belonging to each cluster.

The process will be as follow, an IoT task T_i will arrive to the controller of the fog node. The controller knows which fogs (clusters) are reachable for the

IoT device issuing the task. The controller tries to find the best fog cluster which meets the task's requirements. In case a cluster is found, the controller searches for the best node to execute the task among the nodes belonging to the selected cluster using the optimization problem explained in previous section
This method, which is viewed as a filtering process, reduces the number of devices subject to the selection process.

Next we will be describe the optimization problem (which is similar to the previous one) which will be used to find the best cluster.

The Decision Variables

We define Y_i^k as the decision variable for the second optimization problem. If Y_i^k is 1, it means that cluster k is selected as the best cluster for handling the task i and 0 otherwise as in equation 3.13. When all the decision variables are zeros, it means that none of the reachable clusters are suitable for executing the task. In such a case, based on the scheduling class of the task, if it is set to 1, meaning that it is delay tolerant the decision will be to offload the task to the cloud, otherwise the task will be rejected.

Thus our optimization model has to find the described decision variable which indicates that our problem is a mixed-integer one, as the values that we aim to find are binary ones.

$$Y_i^j = \begin{cases} 1, & \text{if cluster k is selected as best cluster for task i} \\ 0, & \text{Otherwise} \end{cases} \tag{3.13}$$

As our approach is a QoS and security-aware one, the controller has to find the most suitable cluster, that has the ability to satisfy the delay and privacy constraints of the task. As the controller knows the IoT device requesting a task execution, it can know which clusters are suitable for accommodating the task.

Delay Constraint

The task execution process, which starts by issuing a task, finding the best node, submitting the task to the selected node and getting a reply, does not differ from the process described in the previous section. Yet in this part of the second variation we are not concerned with the whole task execution delay which includes the transmission and propagation delay, rather we only care about the processing delay at the selected cluster. Meaning, when we are selecting a cluster and not a node directly we are not aware about the link bandwidth and distance to be able to compute the whole task execution delay. But we still want to make sure that the selected cluster can satisfy the delay constraint of the task, so the estimated execution time in the selected cluster should not exceed the maximum allowed delay, therefor, the difference between both values in equation 3.14 should be greater than zero as in equation 3.15.

$$SpareTime_i^k = MaximumAllowedDelay_i - ExecutionDelay_i^k \tag{3.14}$$

$$SpareTime_i^k \geq \mathbf{0} \tag{3.15}$$

$ExecutionDelay_i^k$ in equation 3.14 is calculated by dividing the instruction count of the task by the average instructions per second of the cluster.

In order to minimize the end-to-end delay we need to maximize the difference between the allowed delay and the estimated processing delay as described in equation 3.15

Privacy Constraint

The controller must select a cluster which should be secure enough to guarantee the minimum security level required by the task and thus not violating the privacy constraint of the issued task request. This means that the difference between the minimum required security level of the task T_i and the average privacy measure of the cluster k as in equation 3.16 should be greater than zero as in equation 3.17.

$$Privacy_i^k = SL_i - AvgPM_k \tag{3.16}$$

$$Privacy_i^k \geq \mathbf{0} \tag{3.17}$$

Similar to what have been discussed regarding the delay constraint, we need to maximize the privacy difference as well to guarantee that the selected cluster satisfies the security level required by the task.

Objective Function

Putting all of this together, our aim is to find a cluster that might contain the best fog node that is suitable to execute the requested task. For that, our objective function is to check if there exit a cluster that maximizes the sum of the factors computed using equations 3.14 and 3.16.

Our objective function is formulated as follow:

$$\text{maximize} \sum_{k=1}^{K} (\alpha SpareTime_i^k + (1-\alpha) Privacy_i^k) * Y_i^k \quad \forall\, i \in \{1, \ldots, N\} \tag{3.18}$$

The α in equation 3.18 is a hyper-parameter used to give a weight for one factor over the other.

This objective function is subjected to several constraints listed below.
As a first constraint, a single cluster can be choosen. This constraint is formalized in equation 3.19 as follow:

$$\sum_{k=1}^{K} Y_i^k \leq 1, \ \ \forall\, i \in \{1, \ldots, N\} \tag{3.19}$$

Next, the selection should guarantee equations 3.15 and 3.17 as well. For that we added the two constraints presented in equations 3.20 and 3.21 respectively:

$$\sum_{k=1}^{K} (SpareTime_i^k * Y_i^k) \geq 0 \ , \ \ \forall\, i \in \{1, \ldots, N\} \tag{3.20}$$

$$\sum_{k=1}^{K} (Privacy_i^k * Y_i^k) \geq 0 \ , \ \ \forall\, i \in \{1, \ldots, N\} \tag{3.21}$$

Resources Constraints

In addition to satisfying the delay and privacy constraints of the IoT tasks, the selected cluster needs also to meet the resources requirements as CPU and memory of the tasks. This means that when the controller selects a cluster to choose one of its fog node, it has to make sure that the resources required by the task do not exceed the average resources available at that cluster.

For this purpose, equations 3.22 and 3.23 were added as constraints to assure that the resources requirements; which are the CPU and the memory, were satisfied without exceeding the average fog resources.

$$\sum_{k=1}^{K}(AvgCPUk - CPU_i) * Y_i^k \;\geq 0\;, \;\; \forall\, i \in \{1, \ldots, N\} \tag{3.22}$$

$$\sum_{j=1}^{K}(AvgMemoryk - Memory_i) * Y_i^k \;\geq 0\;, \;\; \forall\, i \in \{1, \ldots, N\} \tag{3.23}$$

Where $AvgCPU_k$ and $AvgMemory_k$ represent the average CPU and memory available at cluster k, while CPU_i and $Memory_i$ represent the CPU and memory required by the task T_i

The controller might or might not find a desired cluster. If a cluster was found, the controller examines its fog nodes to either select a fog node, offload the task to the cloud or reject the task using the method described in section 3.5.1. If the controller can not find a suitable cluster, based on the scheduling class the task could either be rejected or sent to the cloud.

3.5 Delay Calculation

In this section we will describe how the total delay $TotalDelay_i^j$ is calculated. This factor is an important one as it helps the controller to check whether or not the fog node being examined satisfies the delay constraint of the task requested by the IoT device.

The end-to-end delay is the time difference between the moment the task has been issued and the end of its execution. As mentioned previously, the controller can examine all the fog nodes reachable for the IoT device or the fog nodes of the selected cluster.

So the fog nodes being tested could either belong to the fog cluster the IoT device directly connected to or to a neighboring cluster. Thus the equation used to calculate the delay depends on the location of the fog node being tested if suitable.

Below we describe different delay calculation scenarios:

- The total delay between the IoT device issuing the Task t_i and a fog node f_j belonging to fog k having the controller 'C_k' , where fog k is the cluster the IoT directly connected to, will be equal to:

  $$TotalDelay_i^j = RD_{atC_j} + Delay_{atf_j} \tag{3.24}$$

  where RD_{atC_k} is the request delay at the controller C_k

- The total delay between the IoT device issuing the Task t_i and a fog node f_j belonging to fog *k'* having the controller '$C_{k'}$' , where fog *k'* is a reachable fog to the IoT device, will be equal to:

  $$TotalDelay_i^j = RD_{atC_k} + RD_{atC_{k'}} + Delay_{atf_j} \tag{3.25}$$

  where RD_{atC_k} is the request delay at the controller C_k which is the controller of the fog the IoT device; issuing the task request, directly connected to, while $RD_{atC_{k'}}$ is the request delay at the controller of the reachable cluster *k'*.

The delay at any controller k or fog node j will be represented by the the end-to-end delay equation given in equation 3.26.

$$RD_{atC_k}\ and\ Delay_{atf_j} = d_{tran} + d_{prop} + d_{proc} + d_{queue} \tag{3.26}$$

Transmission, propagation, processing and queuing delays represent the factors making up equation 3.26. Each factor is computed based on the message *'m'* being transmitted, its characteristics and of course the source '*s*' and the destination '*d*'.
The calculation of those factors will take place as follow:

Transmission Delay

$$d_{tran} = Bytes/BW_{s,d} \tag{3.27}$$

Where *Bytes* is the size of data in the message m, while $BW_{s,d}$ is the bandwidth between source *s* and destination *d*.

Propagation Delay

Propagation delay is simply dividing the distance between the source *s* and destination *d* by the propagation speed of the medium as in equation 3.28.

$$d_{prop} = distance_{s,d}/PD_{s,d} \tag{3.28}$$

Where $distance_{s,d}$ is the distance between source *s* and destination *d*, while $\mathrm{PS}_{s,d}$ is the propagation delay between *s* and *d*.

Processing delay

The processing delay for the task execution will be equal to the division of the instruction count of the task over the instructions per second attribute of the node (controller or a fog node) as in equation 3.29.

$$d_{proc} = IC/IPS \tag{3.29}$$

Where *IC* is the instruction count for task i, and *IPS* is the instruction per second.

Queuing Delay

The queuing delay is the sum of the processing delays of all the tasks present at the fog node or the controller. Where n in equation 3.30 is the number of tasks present in the queue at the node.

$$d_{queue} = \sum_{i=1}^{n} d_{proc} \tag{3.30}$$

Chapter 4

Performance Evaluation

In this chapter we describe the simulator used to evaluate the performance of the proposed variations. In this context, we describe the simulator setup that has been done in order to deploy the proposed approach. We also describe the topology and messages creation along with all the parameters being set. Last but not least, we describe the results obtained from the simulation.

4.1 Yet Another Fog Simulator (YAFS)

To implement and analyze our proposed variations, we used the Yet Another Fog Simulator (YAFS) [17]. YAFS is a discrete event simulator designed to analyze the fog applications and the strategies used for their placement, scheduling. Moreover, this simulator provides tools to analyze the routing strategies as well. This simulator is built based on the Simpy library, which is a Python library containing functions that define processes and shared resources.

YAFS is defined by its six main classes which are the core, topology, selection, placement, population and the application class. The core is the class responsible for managing simulation execution to control the life cycle of the processes running.

The topology class, as its name says, it is used to create the fog nodes, cloud servers, sensors and actuators. When defining those network modules, the node's RAM, cost and instruction per simulation time must be provided. Moreover, each node will be represented by its id. This class is also used to define the links between the network devices, which will be defined by the source, destination, bandwidth and propagation delay. This class is accessed by other classes through the core class, since the topology class is a main element of it.

The application class defines modules that run services and messages. Thus using this class we can define the messages' behaviour of any type of application. The message is defined by the size of its data, and the number of instructions.

The selection, placement and population classes are the classes used and modified by user to define the scheduling, routing, resource allocation, and application's deployment in the created topology.

In addition to those six classes, one important class is used which is the distribution class. This class is essential to allow customized distribution among messages. Some of the distribution methods defined in this class are deterministic distribution, uniform distribution, and most importantly the exponential distribution.
After defining the topology, the application with its messages and deployment, and the messages distribution the simulation can start. The simulator issues two excel files as output. The first excel file shows the name of the application, the flow of messages and the processing time in details; through showing the time emitted, reception time, time it went into the module for execution, and the time

the execution was over. The second excel file is concerned with the routing, it shows how the messages moves from the source to destination and which modules are used as routers for the propagation.
We had to modify the simulator in order to adapt it to the environment needed to deploy our approach and architecture. Indeed, our fog nodes are characterized by the CPU, Memory (RAM), instructions per second and privacy measure. For that, the topology class was changed by adding the new attributes which weren't present for creating a fog node.

As for IoT tasks, which will be represented by the messages from the application class, we had to add the other needed variables as security level, scheduling class, CPU, memory, and maximum allowed delay in addition to to the *Bytes* and instruction count variables which were already declared in the class.

Figure 3.2 shows the messages in our proposed approach. Each type of these four messages must be treated differently, for that, the major changes were done in the core class especially in the "send_message" function present in this class.

The selection class is implemented in a way that helps us customize our task allocation algorithm and thus implement the two variation of our approach in chapter 3. For that, different methods were added to allow the deployment of different messages. For example, the first type of message must be sent to the controller of the fog node the IoT device connected to, while the third message must be sent to the fog node (if found) chosen by the controller.

4.2 Simulation Setup

4.2.1 The Hyper-parameter α

In chapter 3 sections 3.4.1 and 3.4.2 we mentioned having the hyperparameter α that is used to give weight for one factor over the other.

We decided to set the value for this parameter after performing several experiments and then deciding which value permits better results.
For our problem, average delay can be used as the best judgment for the best α value.

We created a small scale topology consisting of five clusters each containing a small number of fog nodes. Each cluster has three IoT devices (termed as sensors in the simulator) directly connected to it. Each device can reach either one or two other clusters. The IoT devices are issuing tasks using an exponential distribution where the rate of issuing tasks per time λ is set to 1.5. This value for λ was noticed as the most suitable one after trying several values. This value allows the generation of different types of messages, where lower values of λ causes the repetition of same types of messages.

We run different simulation experiments using different values of α between 0.1 and 0.9 separately. The simulation was done 6 times for each value, where in each iteration different number of types of messages were used with different simulation time being set(as the number of types of messages is increasing we increased the simulation time to allow higher distribution of various messages). The number of messages being generated in each iteration increased from 130 messages in the first one to reach 400 in the last one. In every iteration and for every α we

alpha	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
average delay	0.5278	0.5235	0.5189	0.5152	0.5154	0.5146	0.5144	0.5144	0.511

Table 4.1: The Variation of the Average delay with Respect to the Alpha Value

calculated the average delay; which is the difference between the time the final reply reaches the IoT device and the time the IoT device issues the task request.

The average delay of all the iterations for all values of alphas were computed and presented in Table 4.1. The table shows that as alpha is increasing the average delay is decreasing. Thus our best alpha will be the one that gives the lowest average delay and that would be $\alpha = 0.9$.

4.2.2 The Topology

To evaluate the performance of our approach, four types of topology were created, each containing 5, 10, 15, and 20 clusters respectively. Figure 4.1 is an example of a topology made up of five fog clusters, each fog cluster is consists of a number of fog nodes, having a single controller and number of IoT devices directly connected to it.
Each cluster in every type of topology has a small number of fog nodes, since as mentioned in [4], small scale fogs would result in better performance metrics. This number of fog nodes ranges between 4 and 8 in our work.

Each fog (cluster) has a range between 3 to 5 IoT devices directly connected to it. Each of these devices could reach between 0 to 4 other fogs. Although the number of IoT devices is not large, yet with respect to the topology being created and the number of fogs in each cluster, this number of devices can issue

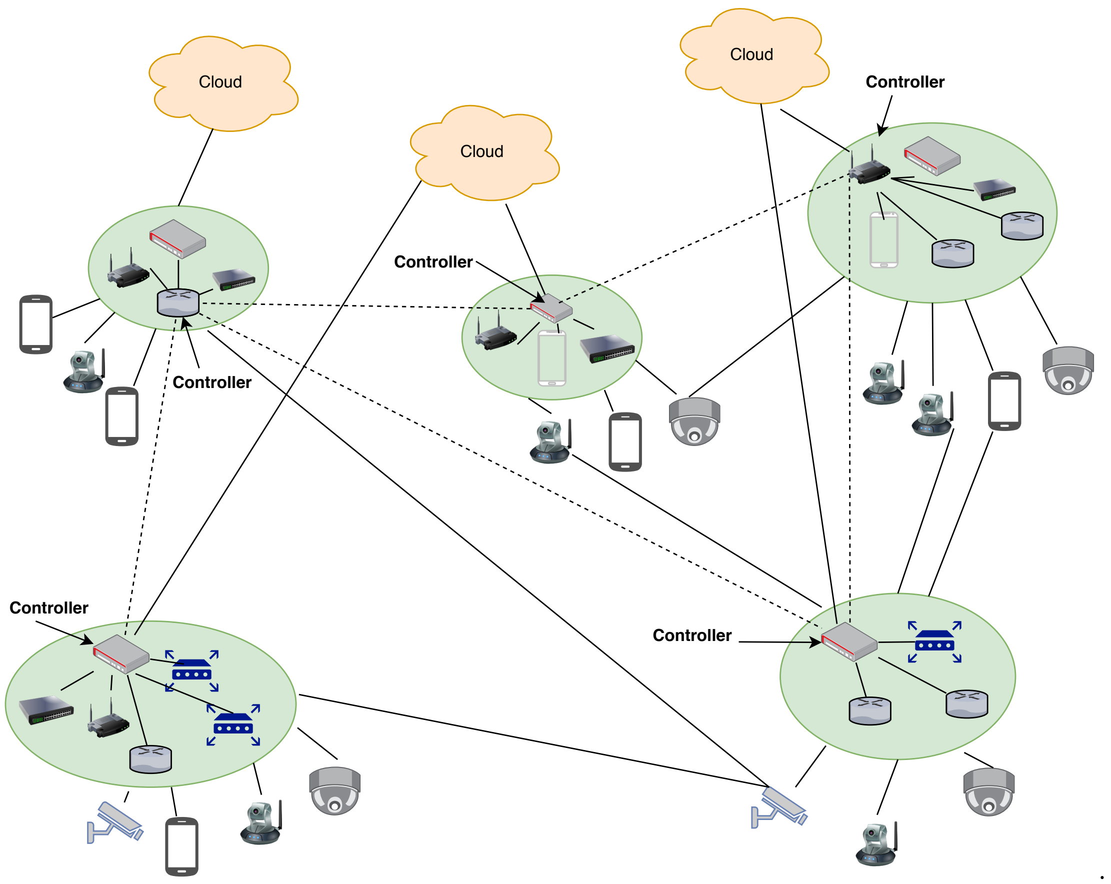


Figure 4.1: Five Clusters Topology

enough tasks in order to be able to evaluate the overall performance. This is due to the fact that the efficiency of the simulation depends on the number of messages being generated and not only on the number of devices issuing the tasks.

To set the characteristics of the fog nodes, we used values from real servers like IBM 7030, and Intel 4004 as stated in [18] and [19] respectively as mentioned in Table 4.2. This was helpful to set a reasonable and synchronized values for the instructions per second, CPU speed and RAM attributes for every fog node.

As for the privacy measure attribute, it was added using a uniform distribution. Meaning, we assumed that each cluster has a range of privacy measures and the privacy measure for every fog node belonging to the cluster is within this range. For example, assume having fog cluster $_k$ with privacy measure range between 0.3 and 0.5; where each value of this range is a trust value generated based on a tool that assess the security strength of fog nodes as described in [16]; so the fog nodes of this cluster will have a privacy measure value in that range.

We assumed having one cloud server acting as the CLOUD. The instruction per second parameter of the cloud must be higher than any value being set to the fog nodes. This is due to the fact that the cloud has higher processing capabilities than any fog node, and for that it's responsible for handling heavy tasks as described in [4].

As for the connections, the bandwidth between an IoT device and a fog node could be either 54 Mbits/s as in wireless 802.11g networks or 100 Mbits/s as in fast Ethernet. The bandwidth between the controllers; which act as routers for IoT devices; and the cloud is set to 10 Gbits/s. While the bandwidth between fog controllers is set to 100 Mbits/s. We adopted these values from the topology created in [20].

4.2.3 The Tasks Creation

Before starting with the simulation we needed to create a large set of tasks. The tasks' requirements as CPU, memory and instruction count were generated using the values set to the fog nodes. For example, the minimum value of cycles per second (CPU) for a fog node in our topology is 740000 as in Intel 4004 [19], while the maximum value is 8696000 as in IBM System/370 Model 158 [21], so the CPU required by tasks will be a value within this range. As for the security level, it was generated uniformly between 0.2 and 0.9. Assuming that lowest trust value being calculated; based on criteria in [16], for a fog node will be 0.2 while the highest value is 0.9.

The scheduling class of every task was also uniformly generated to be either 0 or 1, yet we gave the preference for 0 over 1. We assumed that IoT tasks should not always be delay tolerant; rather the tasks should require resources as fast as possible.

4.3 Simulation Parameters and Values

Table 4.2 summarizes the simulation parameters and their values that were set during the simulation.

Table 4.2: Simulation Parameters and Values

Parameter	Value
Hyper-parameter α	0.9
Task issuing rate λ	1.5
Number of Fog Clusters in each run	5, 10, 15, and 20 respectively
Number of Fog nodes in a cluster	Between 4 and 8
Number of IoT devices connected to a cluster	Between 3 and 5
CPU, Memory and IPS of Fog nodes	As in *IBM 7030,Motorola 68000,Motorola 68020, and Intel 4004*
Privacy Measure of fog cluster	Randomly set between 0.2 and 0.9
Privacy Measure of fog nodes	Uniformly distributed based on privacy measure of cluster the node belongs to
Bandwidth between fog node and IoT device	Either 54 Mbits/s or 100 Mbits/s
Bandwidth between fog controllers and cloud	10 Gbit/s
Bandwidth between fog controllers	100 Mbits/s

4.4 Simulation Scenario

We created five sets each containing 100 different types of tasks. For every set of tasks, the four experiments were performed for each type of topology (which

differs by the number of clusters available). In each experiment, different simulation time was set to allow higher number of tasks being generated as the time increases.
This simulation scenario gives a total of 80 experiments. This scenario was applied for both variations described in chapter 3.

4.5 Results

In this section we first analyze the results for each variation separately for comparison. Moreover, we define the impact of the privacy and security factors on the results by analyzing and comparing the results when including and excluding these factor. Last but not least, both variations will be compared to a baseline approach.

4.5.1 The Flat Based Fog Selection Results

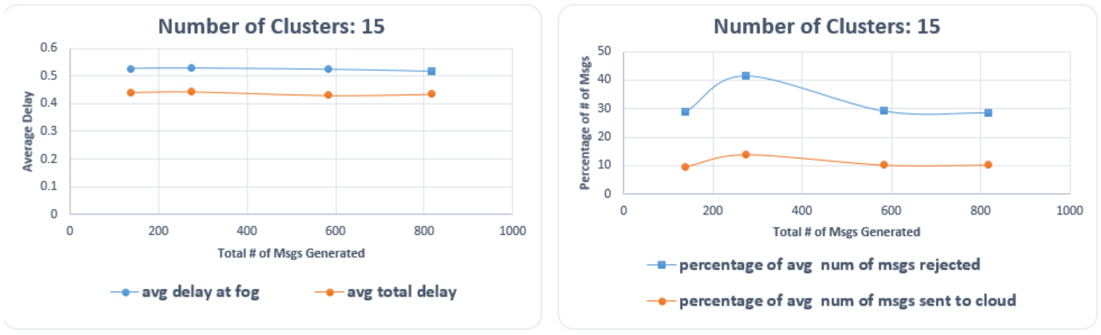
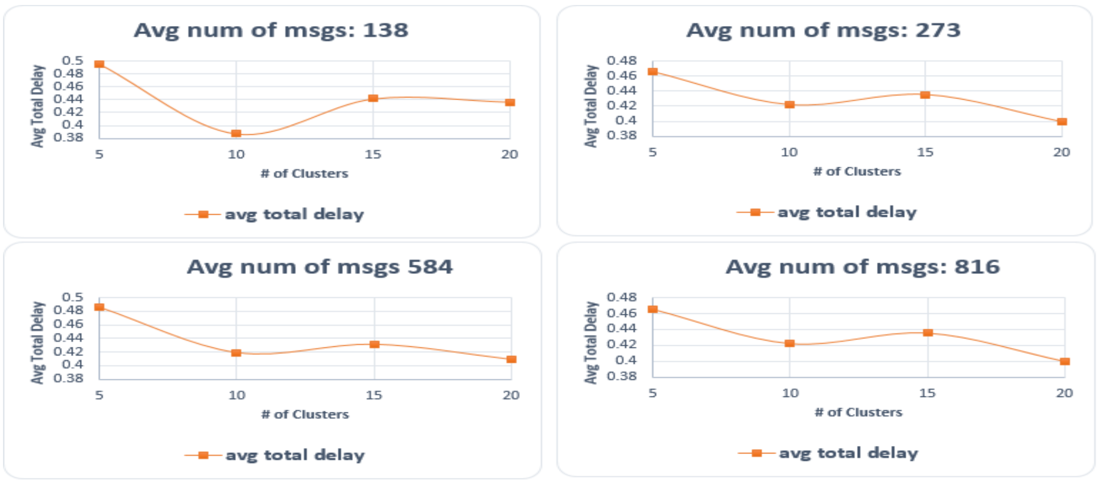
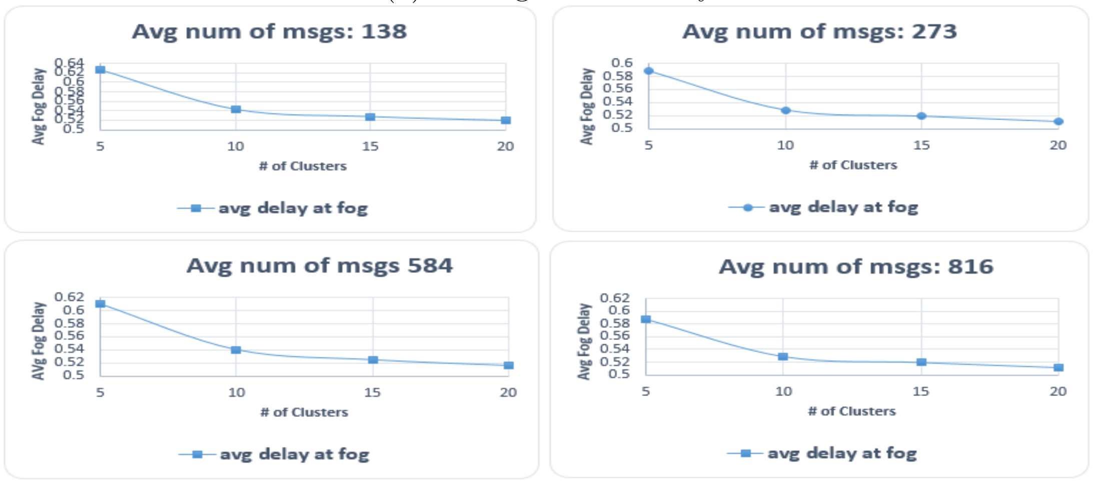


Figure 4.2: The Variation of Values with Respect to Number of Messages

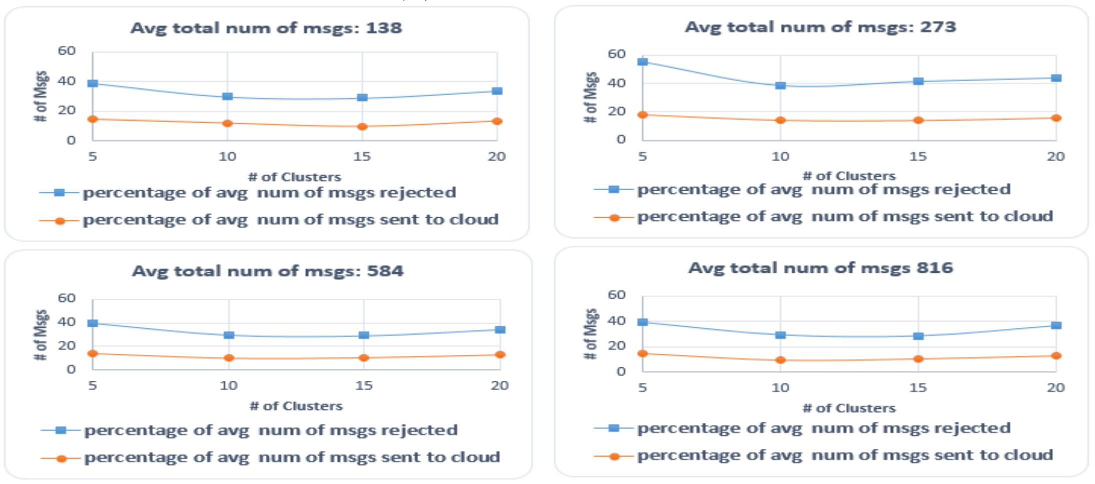
Below we explain the results obtained after deploying the flat based fog selection variation. In this variation, once the controller of the fog cluster receives an IoT task request, it examines all the fog nodes that are reachable to the device



(a) Average Total Delay



(b) Average Fog Delay



(c) Rejected and Offloaded Tasks

Figure 4.3: Flat Based Fog Selection Results

issuing the task using an optimization problem in order to select the best node for execution.

To analyze the results we wrote a script to calculate the percentage of average number of messages being rejected, the percentage of average number of messages offloaded to the cloud, the average overall delay, and the average delay at the fog layer.
Figure 4.2 shows the values obtained for a topology of 15 clusters as the number of messages being generated increases. This figure justifies the slight difference in the values obtained in the upcoming figures for every topology with a certain number of clusters.

Each graph in Figure 4.3 is associated to a specific simulation time where the average number of messages being generated differs. Meaning that in each run of this variation, the simulation time has been increasing in a way to allow the average number of generated messages in the network to increase from 138 messages in the first run to reach 816 messages in the fourth run. This was helpful to determine the impact of the number of messages being generated by IoT devices on the performance of the variation, which shows that the variation behaves the same despite of the number of messages being generated in the network.

Graphs in Figures 4.3 (a) show that the average total delay decreases as the number of clusters available in the network increases from 5 to 20.
Similarly, graphs of Figure 4.3 (b) show that the average fog delay; which is the average delay of messages being executed in the fog layer, also decreases as the number of clusters increases from 5 to 20 clusters.

When the number of clusters increases in the network, each controller will have higher number of fog nodes to take as options into consideration when selecting a node for execution. In other words, when the number of fog nodes available increases, any controller will be able to assign tasks to more fog nodes and thus resulting in lower fog and total delay as shown in Figures 4.3 (a) and (b). Thus, the performance of the variation in terms of fog and total delay hits the objective of the optimization problem described in chapter 3 section 3.5 which is minimizing the delay.

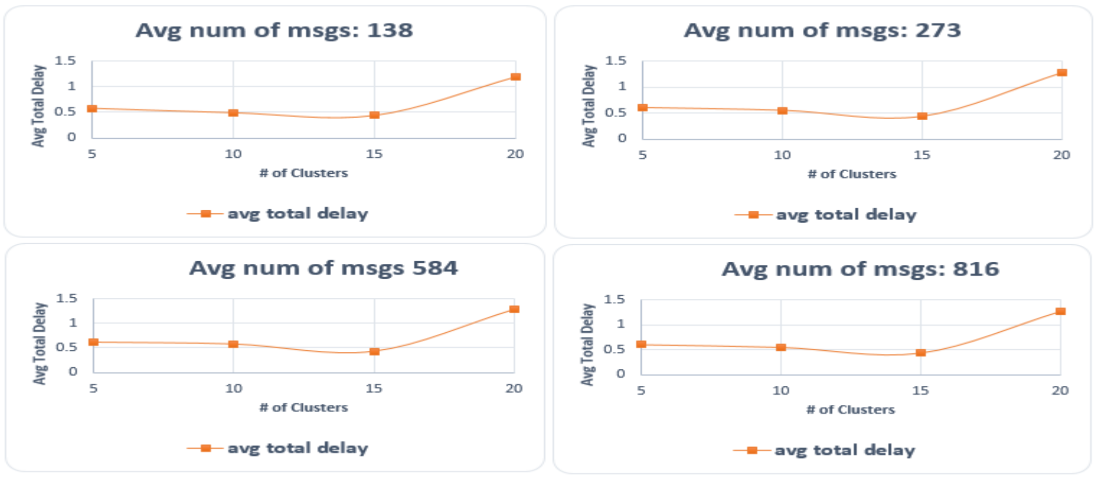
The four graphs in Figure 4.3 (c) show that as the number of clusters increases the average number of messages being rejected and number of messages being offloaded to cloud slightly decreases. This can be justified by the increase in the number of fog nodes that are able to execute the tasks. This increase will lead to a higher probability of task to fog node assignment and thus lower possibility of message rejection and offloading to cloud.

The flat based variation shows that a higher number of fog nodes in fog layer leads to better performance in terms of fog and total delay. Moreover, this increase in number of clusters increases the task to fog node assignments for light weight tasks.

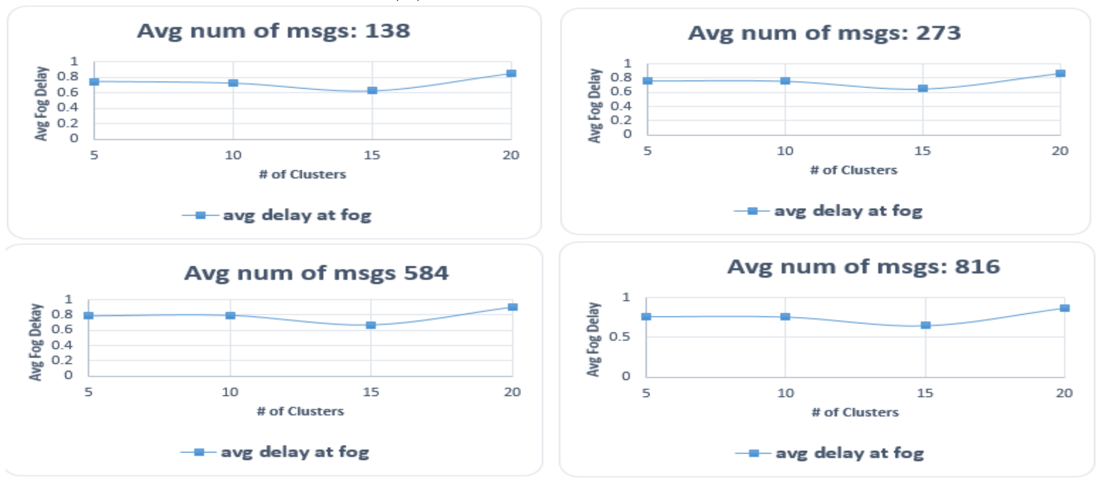
4.5.2 The Clustered Based Fog Selection Results

Below we explain the results obtained after deploying the clustered based fog selection variation. As described in chapter 3 section 3.5.2, in this variation, upon the arrival of a task request at the controller which has a full knowledge about the network, first it has to find a best fog cluster for handling the task and then selects the best fog node belonging to the selected cluster to execute the task.

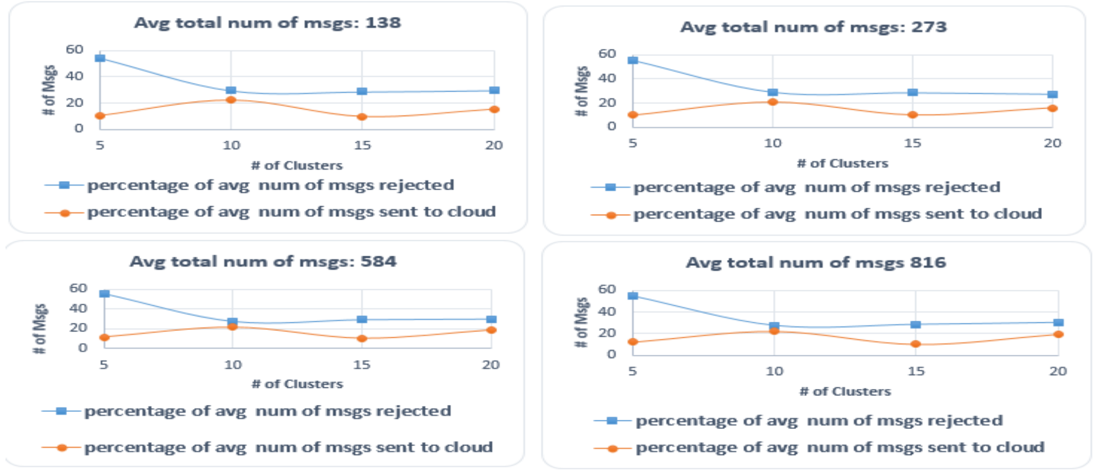
The results were analyzed based on the calculated percentage of average number of messages being rejected, the percentage of average number of messages



(a) Average Total Delay



(b) Average Fog Delay



(c) Rejected and Offloaded Tasks

Figure 4.4: Clustered Based Fog Selection Results

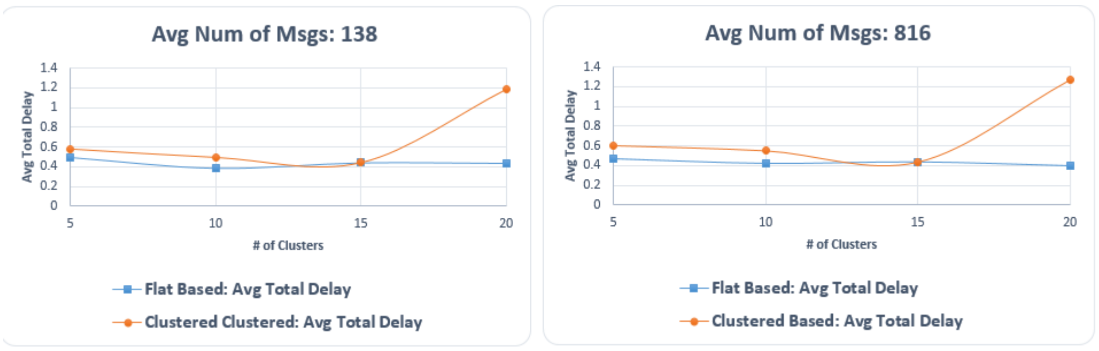
offloaded to the cloud, the average overall delay, and the average delay at the fog layer.

The difference between the graphs of each sub-figure in Figure 4.4 is the average total number of messages being generated. The four graphs of each sub-figure represents the results of the four runs that were performed on this variation. In each run the simulation time was increased to allow the IoT devices to issue as much as possible task requests. The aim of performing four runs for the same variation was to show that the variation permits the same results for every performance metric(average total delay, average fog delay, number of rejected and offloaded messages) even with the variation in the number of messages generated in the network.
The graphs of Figure 4.4 (a) show that the total delay decreases as the number of clusters increases to reach 15 clusters, then it sharply increases as the number of clusters reaches 20.
Similarly, the graphs of Figure 4.4 (b) show that the average fog delay which is the delay of tasks being executed in the fog layer, starts to decrease as the number of clusters in the network increases to reach 15, then it increases when the number of clusters increases to reach 20.
The decrease in the total and fog delay when the number of clusters in the network is between 5 and 15 is justified by the increase in the number of fog nodes that the task could be assigned to, which allows the distribution of arriving IoT tasks on higher number of fog nodes. Yet the increase in the total delay as the number of clusters reaches 20 is due to the increase in the number of clusters that the controller takes into consideration as options to find the best fog clusters for handling the arriving task request.
Graphs of Figure 4.4 (c) show that the percentage of number of tasks being rejected and offloaded to the cloud decreases as the number of clusters available in the network increases from 5 to 20. This is due to the increase of the possible tasks to fog nodes assignments, as an increase in the number of clusters gives

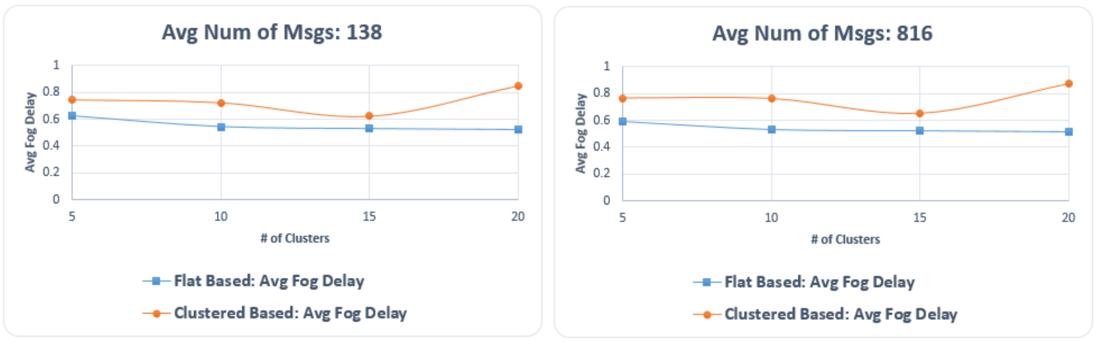
higher probability of finding a best fog cluster for handling a task request by the controller.
The results show that the clustered variation permits better results in terms of average fog and total delay when the number of clusters in the network do not exceed 15 clusters. Meaning that this variation is applicable to minimize the total and average delay when having an average scale fog layer.

4.5.3 Comparing Results of Flat and Clustered Variations

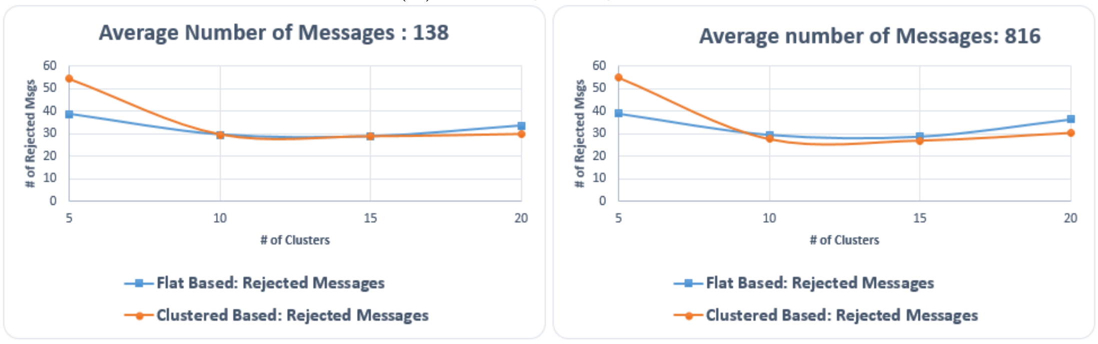
Below we compare the results of the flat and clustered base fog selection variations in terms of average fog and total delay and number of tasks being rejected and offloaded to cloud.
The comparison was done on the first and fourth run of every variation since as mentioned previously, the four runs show that both variations behaves similarly despite the number of messages being generated in the network. Thus the comparison was done when having low average number of tasks(around 138 messages) and high average number of tasks(around 816).
Graphs of Figure 4.5 (a) show that both variations permits the same results in terms of average total delay when the number of clusters in the network is 15, while the flat variation permits better results otherwise.
Graphs of Figure 4.5 (b) show that a flat based fog selection variation is better in terms of fog delay than a clustered based fog selection variation as it gives lower values as the number of clusters in the network increases. This difference is due to the methodology followed to find the best node for execution, where in a clustered based fog selection variation the controller will have higher number of clusters to take into consideration when searching for a best cluster for handling the task as the number of clusters increases. This increase will definitely leads to a higher average fog and total delay.
Graphs of Figure 4.5 (c) show that both the flat and clustered variations lead to



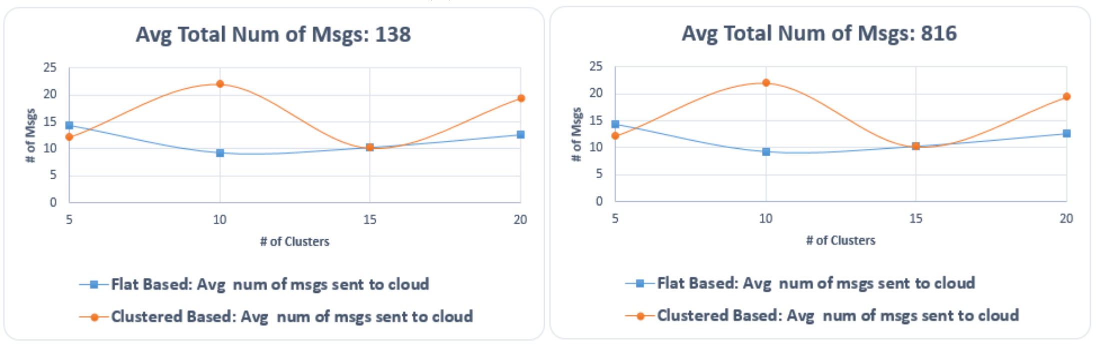
(a) Average Total Delay



(b) Average Fog Delay



(c) Rejected Tasks



(d) Offloaded Tasks to Cloud

Figure 4.5: Flat Based vs Clustered Base

a decrease in the number of rejected tasks as the number of clusters available in the network increases, yet the clustered based variation gives lower values when the number of clusters reaches 20.

Graphs of Figure 4.5 (d) show that in the flat based fog selection variation there was lower number of tasks being offloaded to the cloud than in a clustered based variation when the number of clusters available was 10 and 20. While both variation gives the same results when the number of clusters is 15. In the clustered variation the selection process is based on the average values of the fog clusters capabilities as explained in chapter 3 section 5.2, which means there is a higher probability of not finding a suitable cluster for handling the tasks if none of the available cluster can satisfy the tasks requirements. In such a case the controller can either offload the task to the cloud if it can tolerate delay or reject it otherwise. This explains the higher values of tasks to cloud offloading in the clustered based variation.

As a conclusion for this comparison, the flat based variation is more applicable when having a large scale fog topology consisting of 20 or more clusters, as this variation gives lower values for the fog and total delay as desired and decreases the number of tasks being propagated to higher layers(cloud). On the other hand, when having an average scale topology of around 15 clusters both variation behaves somehow the same and thus both are applicable.

4.5.4 Impact of Privacy and Security Factors

As mentioned previously, we aim on defining the impact of our security aware approach by stating the effect of including and excluding the privacy and security factors from the formalized optimization problem in both variations. In this section we compare the results for each variation separately when the mentioned factors were included and excluded respectively.

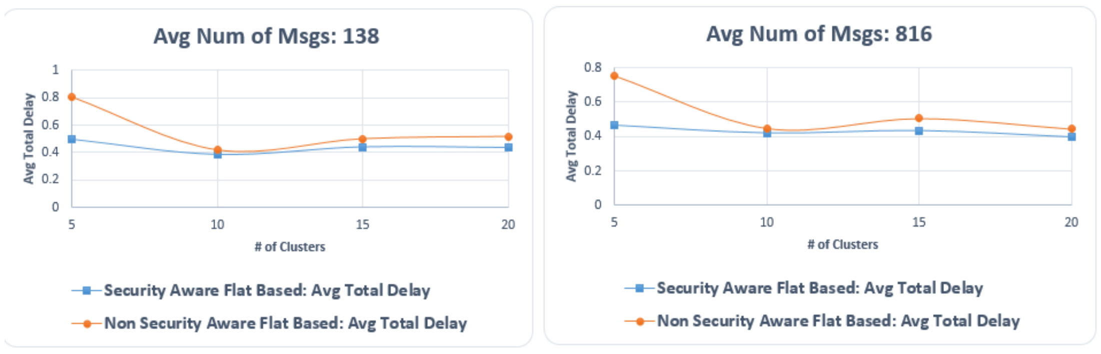
Flat Based Fog Selection: Security Aware vs Non Security Aware

Graphs of Figure 4.6 (a) show that including the security and privacy factors results in lower average total delay. Same goes to the average delay at the fog layer, where lower values were obtained when the factors were included as shown in Figure 4.6 (b).
Graphs of Figure 4.6 (c) show that excluding the security and privacy factors from the formalized problem leads to lower number of messages being rejected. Similarly, lower number of tasks were offloaded to the cloud when security and privacy factors were excluded as well as shown in graphs of Figure 4.6 (d).
The results were expected since excluding these factors means that the fog node selection process has lower number of constraints that need to be met, consequently there will be higher number of tasks to node assignments which leads to higher average total and fog delay.

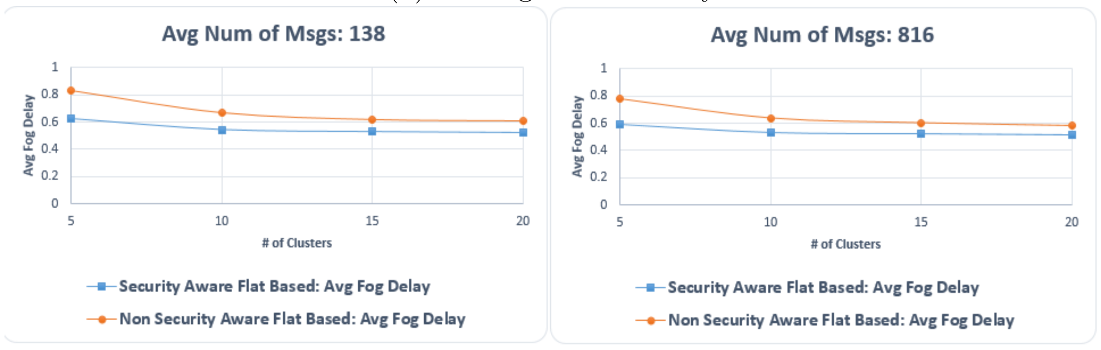
Clustered Based Fog Selection: Security Aware vs Non Security Aware

Graphs of Figure 4.7 (a) show that excluding the security and privacy factors leads to higher average total delay. Figure 4.7 (b) shows that there is a major difference in the average fog delay when the security and privacy were included and excluded respectively, where excluding the factors gives a much higher fog delay. Graphs of Figure 4.7 (c) show that we obtain lower number of messages being rejected when excluding the security and privacy factors from the formalized problem.
While, excluding these factors is leading to a higher number of tasks being offloaded to the cloud as shown in graphs of Figure 4.7 (d).

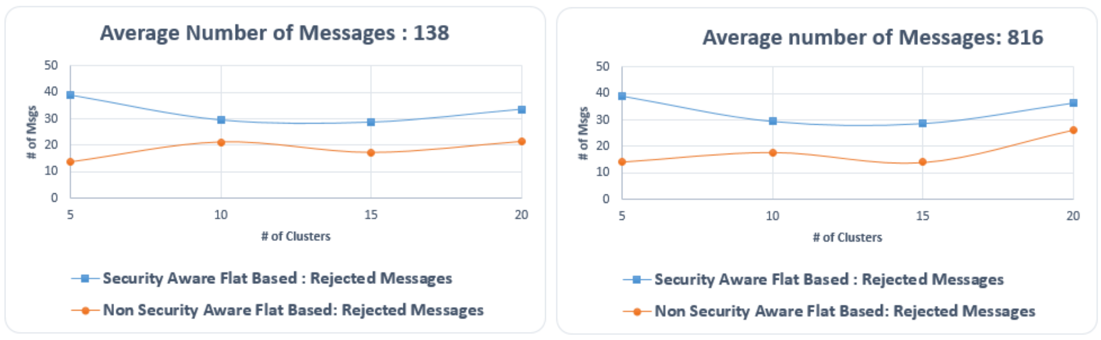
This shows that excluding the factors increases the probability of offloading



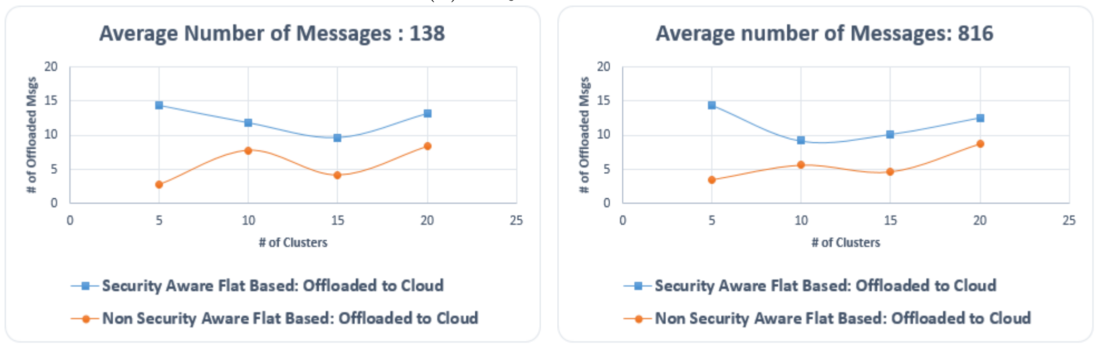
(a) Average Total Delay



(b) Average Fog Delay

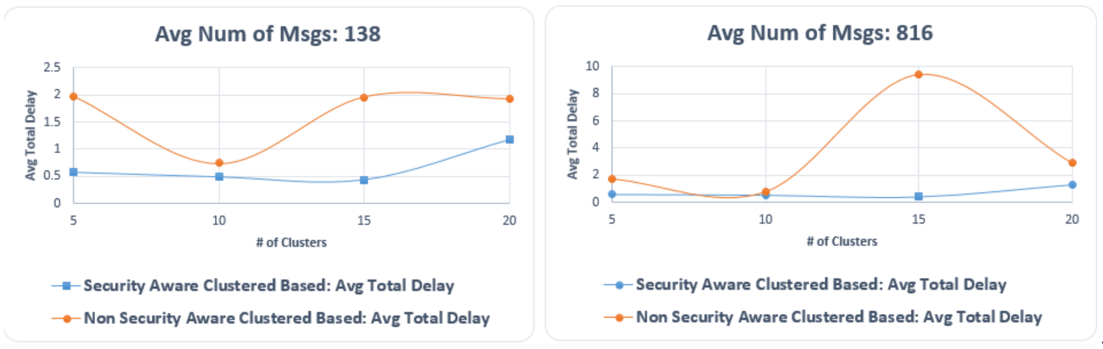


(c) Rejected Tasks

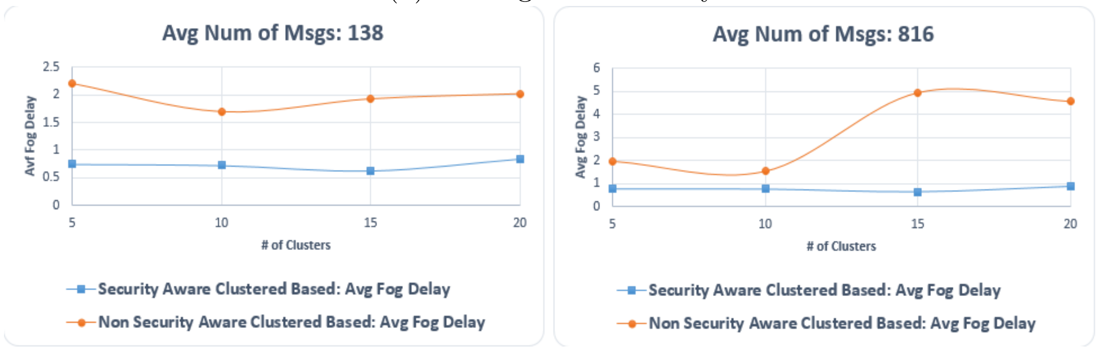


(d) Offloaded Tasks to Cloud

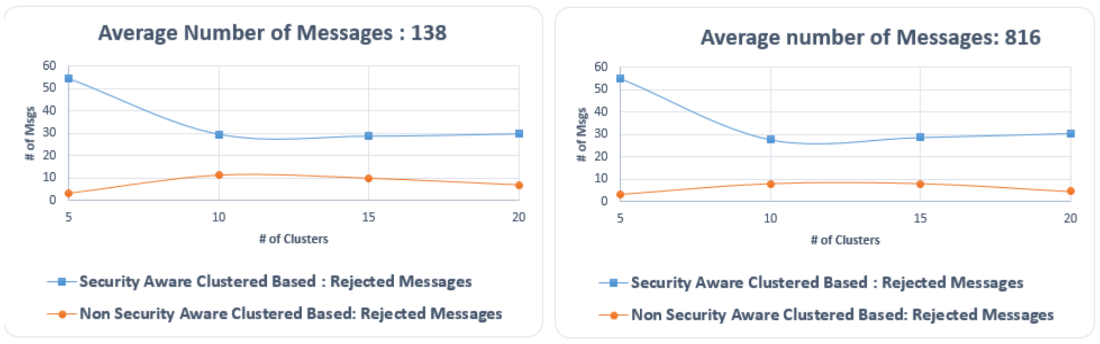
Figure 4.6: Flat Based Fog Selection: Security Aware vs Non Security Aware



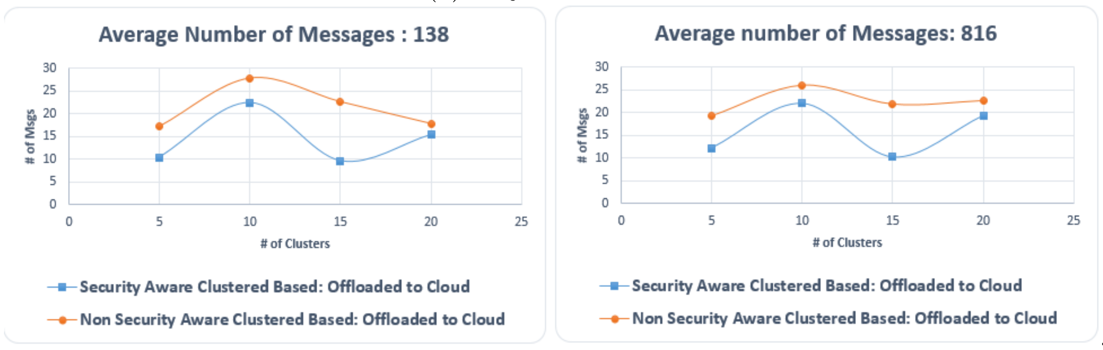
(a) Average Total Delay



(b) Average Fog Delay



(c) Rejected Tasks



(d) Offloaded Tasks to Cloud

Figure 4.7: Clustered Based Fog Selection: Security Aware vs Non Security Aware

a task to the cloud which leads a decrease in the number of tasks being rejected and thus higher delays.

The security and privacy factors leads to higher number of constraints that needs to be met when selecting a node for executing an IoT task. If these constraints were not met during the fog node selection process, the controller could either offload the task to cloud if it can tolerate such a propagation in terms of the delay or reject the task otherwise. This explains the high number of tasks being rejected or offloaded to the cloud in Figures 4.6 and 4.7 when the privacy and security constraints were included in the selection process.

We can conclude that a security aware variation leads to a higher probability of task rejection or task propagation to higher layers since it adds more constraints that need to be met. Yet this type of variation hits the objective which is selecting a fog node that can satisfy the security required by the task request.

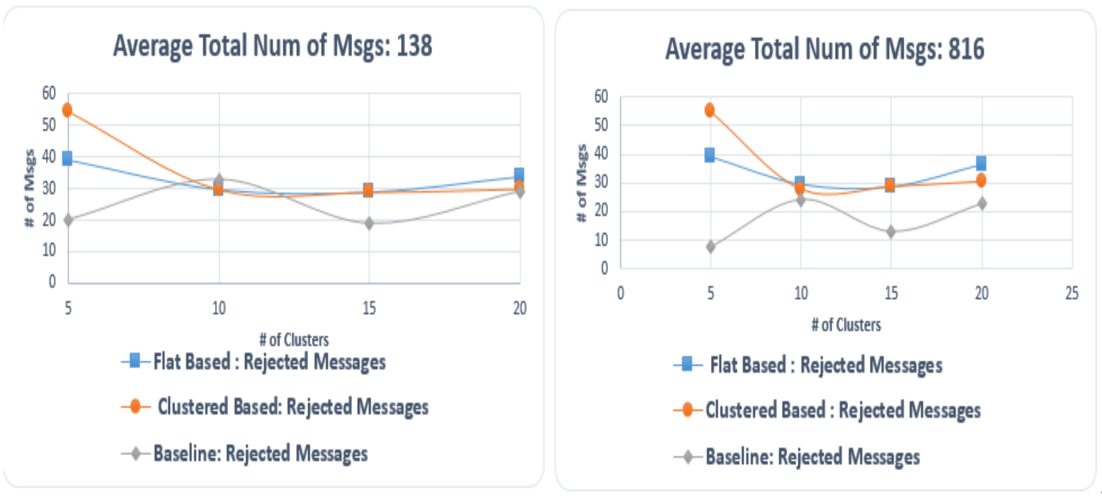
4.5.5 Both Variations Compared to a Baseline

In the baseline approach, when an IoT task arrives at the fog controller, the latter has to find the first available(free) fog node which has sufficient resources as CPU and memory to execute the task.
This approach does not take into consideration the delay and security requirements of the task.

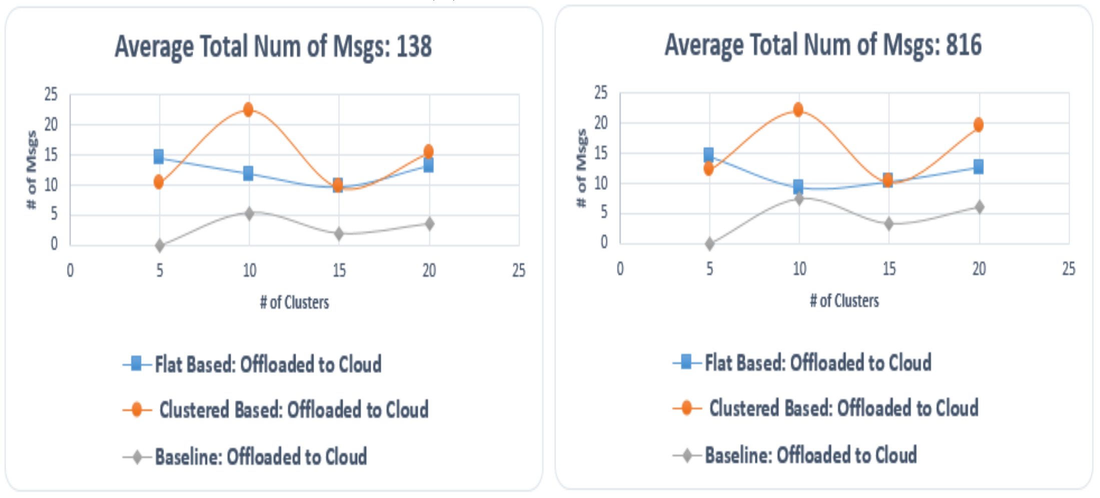
Graphs of Figure 4.8 (a) show that the baseline approach results in lower number of tasks being rejected compared to the flat and clustered based variations.
Same goes to the number of tasks being offloaded to the cloud, where the flat and clustered variations results in higher values compared to the baseline approach as shown in Figure 4.8 (b). This can be explained by the number of constraints available in the baseline approach which are only the resources as in CPU and

memory. Whereas in the other two variations of our proposed approach there exists other constraints like delay and security which were excluded from the baseline approach.

As for the fog and total delays, graphs of Figures 4.9 and 4.10 show that the baseline approach results in a massive total and fog delay when compared to the results obtained from the clustered and flat based variations. This is mainly due to omitting the delay constraint from the baseline approach which does not take into consideration the tasks tolerance for the delay.

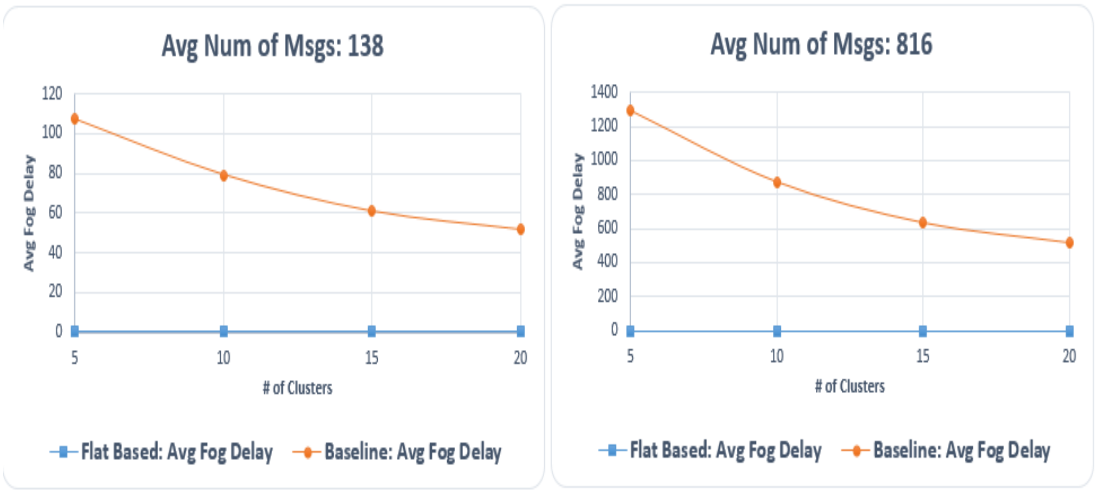


(a) Rejected Tasks

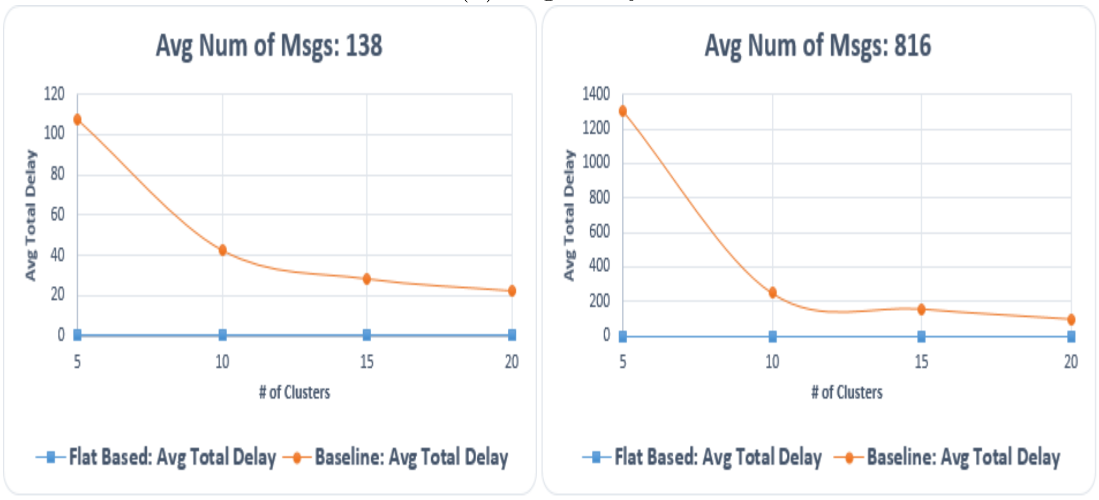


(b) Offloaded Tasks to Cloud

Figure 4.8: Baseline Vs Both Variations: Rejected and Offloaded Tasks

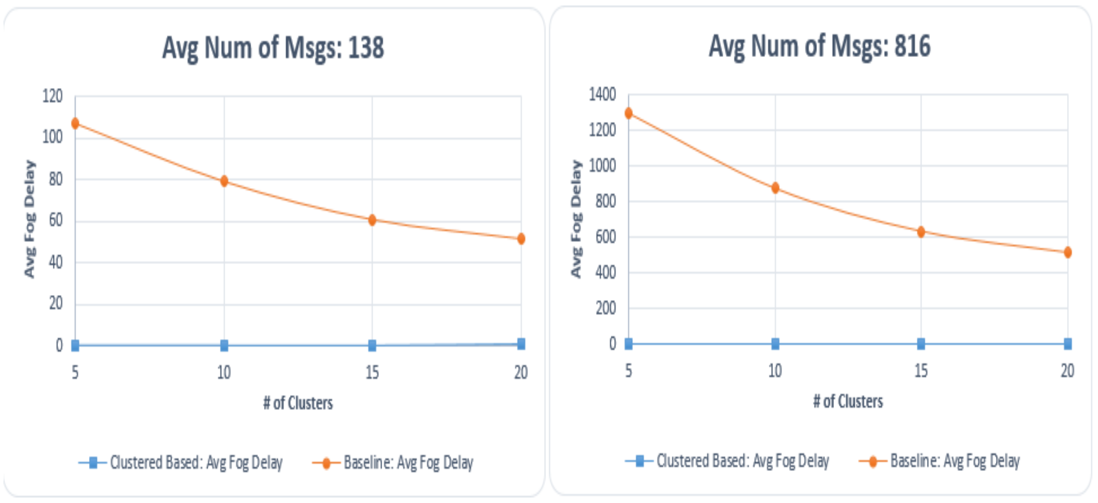


(a) Fog Delay

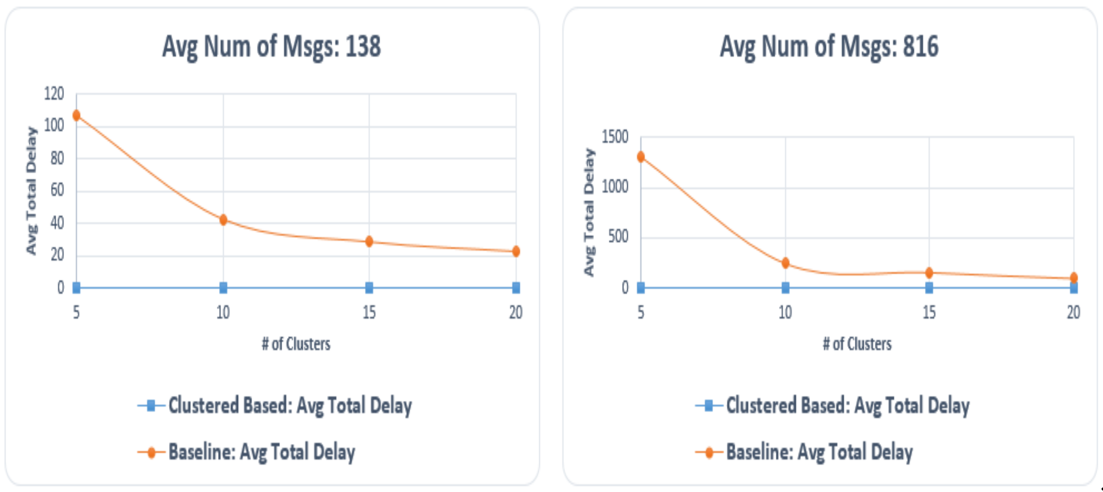


(b) Total Delay

Figure 4.9: Flat Based Variation vs Baseline



(a) Fog Delay



(b) Total Delay

Figure 4.10: Clustered Based Variation vs Baseline

Chapter 5

Conclusion

In our work we proposed two run time resource allocation variations, a flat based fog selection and a clustered based fog selection, which are meant to assign IoT tasks to either fog nodes, cloud, or reject the task if none of its QoS requirement(delay and privacy) are going to be met. We adopted the clustered fog topology as the network architecture for the fog layer where the layer consists of fog clusters each containing a number of fog nodes. Each cluster has a single controller which is responsible to take the task to fog node assignment decision. The cluster will have a number of IoT devices connected to it, where these devices submit their task request to the controller which has to notify the issuing device with the node selection decision.
The main difference between both variations is the procedure that the controller follows to take such a decision. In the flat based fog selection variation, the controller, upon receiving a task request, examines all the fog nodes that are reachable for the IoT device issuing the task in order to find the node that has sufficient resources(as CPU and memory) and can satisfy the QoS requirements of the task. If the controller can not find such a node, then it decides to either offload the task to the cloud if the issued task can tolerate latency or reject it otherwise.

While in the clustered based fog selection variation, the controller, which has the full knowledge about the network, first has to find the best fog cluster that can handle the task based on the clusters' capabilities in terms of resources and whether it can satisfy the QoS requirements of the task or not. If a cluster is found, then the controller next has to find the best node for task execution among the fog nodes belonging to the selected cluster. Task rejection or task offloading to the cloud decisions will take place if the controller was not able to find a node for execution.

For evaluation, we used Yet Another Fog Simulator to implement our variations and perform the simulation. To asses the clustered fog layer topology, our variations were implemented on different scales of such a topology; a topology with either 5, 10, 15 or 20 clusters. The simulation for every variation on every topology scale was performed four times, where in each run higher simulation time was set, which means higher number of tasks being generated in the network. The evaluation was done based on the average total delay, average fog delay, number of tasks being rejected, and number of tasks offloaded to the cloud. The results show that the flat based variation permits better results as the number of clusters in the network increases. While the clustered based variation permits better results when having an average of 15 clusters in the network.
Moreover, we were able to asses the security feature by including and excluding the security and privacy factors from the formalized problems. The results show that including the factors leads to higher number of tasks being rejected since the constraints that need to be met is higher, yet these factors guarantee satisfying the security requirements of the tasks.
Last but not least, our variations were compared to a baseline approach. The applied baseline approach aims on finding the first available fog node with sufficient resources to execute the task without taking into consideration the delay and security requirements of the task.
The results show that the baseline approach results in lower number of messages

being rejected and offloaded to the cloud since the number of constraints, like security and privacy, needed to be met are omitted. While the flat and clustered based variations results in lower fog and total delay compared to the massive delays resulting from the baseline approach.

Bibliography

[1] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource provisioning for iot services in the fog," in *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*, pp. 32–39, IEEE, 2016.

[2] B. Varghese, N. Wang, D. S. Nikolopoulos, and R. Buyya, "Feasibility of fog computing," *arXiv preprint arXiv:1701.05451*, 2017.

[3] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, pp. 37–42, ACM, 2015.

[4] L. Peng, A. R. Dhaini, and P.-H. Ho, "Toward integrated cloud–fog networks for efficient iot provisioning: Key challenges and solutions," *Future Generation Computer Systems*, vol. 88, pp. 606–613, 2018.

[5] N. Mostafa, I. Al Ridhawi, and M. Aloqaily, "Fog resource selection using historical executions," in *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 272–276, IEEE, 2018.

[6] H. A. M. Name, F. O. Oladipo, and E. Ariwa, "User mobility and resource scheduling and management in fog computing to support iot devices," pp. 191–196, 2017.

[7] J.-B. Wang, J. Wang, Y. Wu, J.-Y. Wang, H. Zhu, M. Lin, and J. Wang, "A machine learning framework for resource allocation assisted by cloud computing," *IEEE Network*, vol. 32, no. 2, pp. 144–151, 2018.

[8] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," pp. 877–880, 2012.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[10] T. Choudhari, M. Moh, and T.-S. Moh, "Prioritized Task Scheduling in Fog Computing," *Proceedings of the ACMSE 2018 Conference*, pp. 22:1—-22:8, 2018.

[11] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*, pp. 105–110, IEEE, 2015.

[12] V. Kochar and A. Sarkar, "Real time resource allocation on a dynamic two level symbiotic fog architecture," in *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*, pp. 49–55, IEEE, 2016.

[13] B. J. Ko, K. K. Leung, and T. Salonidis, "Machine learning for dynamic resource allocation at network edge," in *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX*, vol. 10635, p. 106350J, International Society for Optics and Photonics, 2018.

[14] J.-y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," *arXiv preprint arXiv:1901.10023*, 2019.

[15] A. T. Nassar and Y. Yilmaz, “Reinforcement-learning-based resource allocation in fog radio access networks for various iot environments,” *arXiv preprint arXiv:1806.04582*, 2018.

[16] R. Shaikh and M. Sasikumar, “Trust model for measuring security strength of cloud computing service,” *Procedia Computer Science*, vol. 45, pp. 380–389, 2015.

[17] I. Lera, C. Guerrero, and C. Juiz, “Yafs: A simulator for iot scenarios in fog computing,” *arXiv preprint arXiv:1902.01091*, 2019.

[18] Wikipedia contributors, “Ibm 7030 stretch — Wikipedia, the free encyclopedia,” 2020. [Online; accessed 27-May-2020].

[19] Wikipedia contributors, “Intel 4004 — Wikipedia, the free encyclopedia,” 2020. [Online; accessed 27-May-2020].

[20] A. Yousefpour, G. Ishigaki, and J. P. Jue, “Fog computing: Towards minimizing delay in the internet of things,” pp. 17–24, 2017.

[21] Wikipedia contributors, “Ibm system/370 — Wikipedia, the free encyclopedia,” 2020. [Online; accessed 27-May-2020].